ACTIVE LEARNING BASED ON A HYBRID NEURAL NETWORK MODELLER

Yonglong Wang

PhD

1998

University of Abertay Dundee

ACTIVE LEARNING BASED ON A HYBRID NEURAL NETWORK MODELLER

A thesis submitted in partial fulfilment of the requirements
of the University of Abertay Dundee
for the degree of Doctor of Philosophy

by

Yonglong Wang

University of Abertay Dundee
Dundee, United Kingdom

June 18, 1998

I certify that this thesis is the true and accurate version of the thesis approved by examiners.

Signed  Date 11 August 1998

Director of Studies

ABSTRACT OF THE PHD THESIS

Various methods are investigated for selecting training data for the purpose of training neural networks. A new method called MIQR (Maximum Inter-Quartile Range) is proposed for effectively selecting a concise set of training data. In addition, the *ensemble* concept is introduced in this new method. Data selection is not unduly influenced by *"outliers"*, rather, it is principally dependent upon the *"mainstream"* output of the ensemble networks. Encompassed in the new method is a very simple ancient Chinese philosophical idea, *i.e.* "the *minority obeys the majority*". These techniques are nonparametric in the sense that several different neural networks comprise an ensemble or committee and co-operatively work together with each other to achieve a common goal. Because these are different neural networks (hybrid model), they can be complementary in the entire learning system, and therefore effectively enhance the entire learning system's efficiency and accuracy. For learning, the neural networks attempt to actively select the most informative and important training data.

The methods described in this thesis pleasingly satisfy this need, and compare favourably with contending methods. Many experiments have been done to corroborate theoretical and empirical conjectures. The results are quite pleasing in that this new method is not only as "active learning" much better than "passive learning" both in data selection and in generalisation performance, but also outperforms other existing contending active learning methods. In particular, the results are very satisfying and interesting when the method is applied to discontinuous functions. Although the experiments are conducted with clean data selection, it should be easy to extend them to noisy data selection since the method developed is validated using unlabelled data. The algorithm developed for these methods has been rigorously tested, and proves to be highly autonomous and robust. The methods developed here are not restricted to use on neural networks. More generally, they can be applied to other scientific research and economic fields, even educational and sociological behaviour.

ACKNOWLEDGEMENTS

First of all, I would like to thank Dr Bill Samson, Dr David Ellison and Dr Louis Natanson for their supervision.

I would like to thank Ms Pat I Dugard with whom I have benefited from discussion.

I would like to thank the secretarial staff in the School of Computing. In particular, I thank Mrs Patricia McKelvie for her kind help.

Personally, I heartily express much gratitude to John Ellis and Elizabeth Ellis for their kindness and generosity. This couple provided much help when I lived in Scotland, and helped me feel more comfortable and warm in an unfamiliar environment.

I take this opportunity to thank my father and my mother for bringing me up. Their hard work and sacrifice for their children will be always my power for struggling and going forward for life.

Table of Contents

Glossary and Definitions

1. *Cycle*: one pass through a training set for an individual network is called a cycle. The training of a network on a fixed training set to a given threshold will in general require many cycles.

2. ***Feature space***: it is also referred as to vector space. For a particular problem domain, feature space is usually composed of multi-dimensional vectors. For example, in neural network ensemble, the feature space is composed of networks of the ensemble in the function space.

3. ***Early-stopping criterion***: A technique which is used to decide when to stop training in order to avoid a network's "over-training".

4. ***Exemplar***: According to the custom of the neural network community, a data point selected by using active learning is referred to as an exemplars.

5. ***Generalisation***: the ability of a neural network to correctly classify new patterns. It usually is measured by the root mean squared error between the output of the network and target output.

6. ***Iteration***: the process of training an entire ensemble of networks before selecting a new exemplar is called an iteration.

7. ***Labelled data and unlabelled data***: In the mappings of $\mathbf{X} \rightarrow \mathbf{Y}$ (here X and Y can be any dimensional space), if $\mathbf{X}$ is known, then $\mathbf{Y}$ is also determined by $\mathbf{Y}=f(\mathbf{X})$. Such data is referred to as labelled data; Otherwise, no knowledge is required of the target function to be approximated, it is referred to as unlabelled data.

8. ***Network complexity***: the degree of complexity of the network architecture is normally the number of hidden units.

9. ***Network learning efficiency***: defined as the ratio of the number of data points resampled to the number of weights of the network, using a method for selecting examples;

10. ***Nonparametric***: parametric statistical procedures are dependent upon rigid assumptions, for example, that samples have been drawn from normally distributed populations with equal variance, or tests based on Student's *t* distribution. In contrast, nonparametric procedures are not concerned with population parameters or sample distributions, being valid under very general assumptions.

11. ***Over-fit***: A network that is not sufficiently complex can fail to detect fully the signal in a complicated data set, leading to under-fit. A network that is too complex may fit the noise, not just the signal, leading to over-fit. Over-fit is especially dangerous because it can easily lead to predictions that are far beyond the range of the training data with many of the common types of neural networks. Over-fit can also produce wild predictions in multilayer perceptrons even with noise-free data.

12. ***Random sample***: data is selected in such a way that all points have an equal probability of being chosen.

13. ***Resampling***: the process of selecting a training data set (no matter whether using an active or passive learning method) is called resampling.

14. ***Resampling efficiency*** is defined as the ratio of the number of data points resampled to the total training iterations, and is a measure of the training procedure's efficiency.

Chapter 1

Introduction

1.1 Aim and Significance

Neural network learning procedures are popular and effective techniques for fitting a nonlinear model to an unknown mapping (i.e. a black box) by learning from a set of "training examples". Such training examples, however, often contain redundant, useless, or contaminated information; in particular, in some circumstances, it is even difficult or dangerous to obtain them. Network learning may result in the following cases: firstly, it may result in nonconvergence; secondly, it may consume unnecessary resource information or training costs; thirdly, it may result in a wrong or misleading output; finally, it may be impossible for the neural network to continue learning because of being unable to obtain necessary training examples. This dissertation includes novel techniques for selecting training examples from a given input space. The techniques developed provide an effective means such that the neural network actively selects its own required training examples, rather than passively accepting training examples. It is found that some particular data points play an important role in the network learning, being the most useful information. How to search for such data points is the key issue to improve network learning's efficiency and accuracy. An extra benefit of active learning is to minimise training examples, consequently reducing the training cost.

In this dissertation, another contribution concerns the ensemble concept. Our study indicates that using network ensembles can give enhanced or improved network learning accuracy and reliability, and a better generalisation performance with active learning. There are a number of practical advantages to either decomposing a task into subtasks or combining several different solutions to the same task. This philosophical idea of the

ensemble is particularly significant in sociological and educational behaviour. An individual force or intelligence is weak, and his idea is often incomplete or inaccurate. On the other hand, a group or collective force is powerful and they can correct or rectify each other's individual disadvantages or inaccuracies. In education, for example, my supervisory team consists of Dr Bill Samson, Dr David Ellison and Dr Louis Natanson, each having his expertise and advantages. They work together and guide me to achieve my PhD Degree. In general individuals can share responsibility and work co-operatively in order to achieve their common goals.

More importantly, we integrate our data selection technique with ensemble methods so that data selection is dependent upon mainstream instead of outliers, reflecting an important law: minority obeys majority. This law is also a general law, standing for all principles of mankind and nature. For example, presidency, enterprise decision making, and life survival.

It has been shown that neural network learning procedures are inherently statistical techniques. In particular, they are a special case of nonlinear nonparametric regression [106], using least-squares as the optimisation criterion. This dissertation employs and extends statistical models of learning. The development is followed through with empirical implementation and testing. The resulting methods are intuitively pleasing and straightforward to compute. Practical implementation issues are addressed in detail, and the techniques are demonstrated on example learning tasks. Therefore, this dissertation should be of interest to the practitioner interested in applying these methods.

In neural network research, active learning and ensemble are two distinct methods. In our study, an ingenious combination of both active learning and ensemble methods provides an effective means of neural network learning. This result is not only applicable to neural network learning, but also applicable to statistics and more importantly, from the standpoint of philosophy, applicable to educational and sociological study.

1.2 Motivation

Traditionally, most approaches to neural network learning simply present the network with all of the training examples or randomly selected examples. In practice, this can be very costly for large training sets because training sets may include many redundant examples. Besides, many randomly selected examples may contain useless information, causing the network to fail to converge to an acceptable generalisation error. Furthermore it may even be infeasible to obtain some training examples due to reasons such as dangerous or uncertain environments. Thus, the neural network itself actively selecting/searching for the most useful information or minimising a set of training examples can be very significant and beneficial in certain environments.

This active behaviour in the neural network research field is referred to as active learning, active data selection, query-based learning, or active sampling technique. According to the custom of the neural network community, we refer to examples selected in this way as "exemplars". Our goal is to minimise the number of exemplars and at the same time not to harm generalisation performance. Generalisation is the ability of neural networks to correctly classify new patterns. We will use the term "exemplar selection" to refer to all techniques for selecting exemplars. The methods we develop for selecting training examples from a given data set are known as "active" selection of training examples. Rather than passively accepting examples produced by some unknown data generating process, active selection can be used to judiciously or intentionally select the exemplars which the network requires most. Therefore, the network is able to actively take part in its own training. As a result it considerably improves the network's learning efficiency.

If this active data selection technique proves to be generally feasible, several benefits are immediately apparent for a wide range of applications. First, if a set of candidate examples is too large to contain in working memory, the method provides a useful solution for selecting a small, yet still sufficiently informative training set. Second, the development of these techniques does not depend upon the type of estimators (neural network in this case); thus they may be used by any statistical estimator. Third, the

method could be used in data-driven learning techniques where the model complexity is directly related to the number of exemplars, for example, the complexity of the basis functions are proportional to the number of exemplars [85], which are chosen randomly from the set of candidates. Fourth, this way of choosing the exemplars could make such algorithms more efficient by locating some particular data points, from which geologists may gain enlightenment. In geological exploration, for example, in oil fields, it is very expensive to determine drilling sites. Finally, query "by committee" provides an effective means of improving generalisation performance.

In general, the aim of active learning is to maximise information gain and minimise generalisation error[97]. Plutowski confirmed that maximising the information gain has the side benefit of making training more reliable and efficient [88]. Because of the minimal training set, the overall cost of training will be reduced, and generalisation can be improved over using all of the available examples as well. Plutowski also points out that the run-time complexity of learning algorithms depends upon the number of examples used in training. This is especially evident in iterative learning algorithms that process each example a large number of times over the course of training. Empirical experience clearly indicates that the accuracy and reliability of network training depends greatly upon the amount of data available for training. In practice, we have seen applications with huge data sets, in which the amount of processing time required for learning is driven by the amount of time necessary for the learning algorithm to iterate through the data set. Formal considerations provide considerable justification for this empirical experience. Much work has been done examining generalisation ability as a function of the number of training examples, in a variety of sources [7][8][69][46][48][63]. Most of these results indicate that for examples passively sampled according to the environmental distribution, the number of examples sufficient for proper training grows with the complexity of the network model. Measures of network complexity include the number of learnable parameters ("weights"), nodes, or layers, or other smoothness conditions on the network function, or the number of bits of precision to fit the network. A common result is that the number of examples sufficient to ensure a small generalisation error of e grows at least as fast as the number of weights, p [88].

Plutowski provides a concrete justification, where it is demonstrated that for a particular common neural network architecture, with reasonable choices of e, the sufficient number of examples is approximately $O(\Delta^4 p \ln\Delta)$, where Δ is a bound on the maximum absolute magnitude of the weights. This gives a typical example of how the number of examples "sufficient" to obtain a desired level of generalisation depends upon the number of network weights, namely, network complexity. In his experiment, he employs a two hidden-layer network architecture to approach more complicated mappings. He also illustrated the rate at which the data requirement grows as network complexity increases. In addition, it has been shown that the benefit of adding a randomly selected example to the training set decreases exponentially as the size of the training set increases[88].

An attempt is made to counteract this dilemma by judicious selection of exemplars by exploiting knowledge gained by learning on previously selected examples based on a set of combining networks with a simple architecture. In practice, the better the generalisation performance needed, the more strict the training threshold must be. On the other hand, a more strict training threshold means more complex neural networks. This work shows that by using ensemble networks, network complexity can be considerably reduced. We can implement any mappings by only one hidden-layer network architecture with a smaller number of hidden units than in previous work such as Krogh [59].

There are two principal methods in active learning, one is Maximum Error and another is Maximum Variance. The former is represented by Plutowski's ΔISB[88] and Sung's EISD [99] and the latter by RayChaudhuri [89] and Krogh [59]. The weakest deficiency of the Maximum Error method is that it can only be applied to model reasonably smooth functions because its algorithm requires the computation of an inverse Hessian matrix consisting of at least first derivatives of the target function to be mapped. Besides, Maximum Error lacks a clear stopping criterion because its algorithm only validates deterministic or labelled data. The Maximum Variance method makes use of the ensemble method to overcome the above deficiencies. However, in practice, it is found that Maximum Variance is frequently affected by local minima and quits training too early or fails to obtain a desired training level. Theoretically, exemplar selection by the

Maximum Variance method is essentially based on the assumption that values used in the computation of variance should be normally-distributed. The frequent presence of outliers in network outputs indicates that this is not the case. As a result, when using the Maximum Variance criterion, the training may be trapped in a "dead valley" because when approaching a particular problem, for example, a discontinuous function approximation, the learning procedure repeatedly locates the same data point and wastes a large number of useless iterations because of the disproportionate influence of outliers on the variance. Therefore, proposing a new effective method for active learning is essential.

1.3 Overview

This section provides an overview of the remainder of this dissertation.

In the neural network community, "active learning" encompasses two distinct techniques for choosing training examples given information derived from a network partially trained on a set of previously acquired examples. One is "active sampling" (searching for new data to be added to the set of available examples) and another is "active selection"(sifting or filtering particular data from an existing set of available examples). In Chapter 2 previous research results on both approaches is reviewed.

In Chapter 3 is a brief introduction to neural networks to illustrate neural network architecture, operation, and training and learning, which are necessary to understand the remainder of this dissertation.

Ensemble concepts are presented in Chapter 4. A theoretical justification is given that ensemble networks improve generalisation performance and the average ensemble mean square error is equal to $1/N$ of an individual one assuming the members of ensemble networks are mutually independent with zero mean. In addition, the trade-off of bias and variance of neural networks is discussed in more detail. Also a number of methods for creation and combination of an ensemble network are presented. In the last section of this chapter, cross-validation and bootstrapping are discussed, since they are the most commonly used techniques for resampling data sets and creation of network ensembles.

In Chapter 5 an algorithm is presented for reducing generalisation error in the approximation of a function using a neural network ensemble. It is confirmed that in order to have improved generalisation, each member of an ensemble network and training examples must be as diverse as possible. Chapter 5 culminates in a demonstration of active data selection using the Maximum Variance criterion, where we refer it to as "discrepancy". The discrepancy is defined as the variance of the output of ensemble members averaged over unlabelled data, so it quantifies the disagreement among the networks.

In Chapter 6, a new method for actively selecting examples is formally developed using ensemble networks. The method for creation of ensemble networks is to use bootstrapping, initial random weights and averaging the outputs of ensemble networks. The procedure starts by selecting a small data set chosen randomly, and trains the ensemble network models with random subsamples of this data set, then the different outcomes of the ensemble models are computed. We propose a new criterion for selecting training examples - selecting the next data point corresponding to the largest inter-quartile range between the upper quartile and lower quartile. Intuitively, sampling techniques seek training examples which are "representative", in some sense, of the space of all possible training examples. Since a set of ensemble networks is used to approach the same learning task, clearly, outputs of the members of the ensemble are not identical. The ones furthest away from the "majority" may be "unrepresentative" or "outliers" and should be pruned so as to ensure a general trend to the desired target output. After pruning the "outliers" of the ensemble networks, we compute the inter-quartile range of quartiles. The points which have large inter-quartile range are the "exemplars" where more information is required for learning. In this way, the ensemble network models allow us to estimate the change in inter-quartile range for each candidate example. We simply pick the example that has the largest inter-quartile range. We then fit ensemble models to the new sets of exemplars, and repeat until we achieve an acceptable error level.

Reviewing previous active learning methods, such as Maximum Error and Maximum Variance, it is found that Maximum Error not only is computationally complicated and expensive, but also requires the assumption that the function to be approached is reasonably smooth because these Maximum Error methods have to compute the inversion of a huge Hessian matrix such as Plutowski's Δ ISB and Sung et.al's EISD (expected integrated squared difference). It is unable to approach a sharp discontinuous function such as the square wave, because we cannot compute the Hessian matrix. In addition, Maximum Error only uses a single network, therefore it cannot make use of advantages of ensemble networks. Furthermore, Maximum Error only approaches labelled data. Another contending method is Maximum Variance, which is very popularly used in ensemble network learning. However, in practice, it is found that the Maximum Variance method frequently traps into a local minimum and quits training too early or training cannot go forward. In addition, when approaching a sharp discontinuous function it will considerably reduce the resampling efficiency and the learning efficiency.

The approaches developed here are directly derived from nonparametric statistics, and highly differentiated from previous works. In the neural network community, most solutions of objective criteria for active learning use Maximum Variance or Maximum Error, or variations on them. Ideally, values used in the computation of variance should be normally-distributed. The frequent presence of outliers in network outputs indicates that this is not the case. As a result, when using the Maximum Variance criterion, the training may be trapped in a "dead valley" because when approaching a particular problem, for example, a discontinuous function approximation, the learning procedure repeatedly locates the same data point because of the disproportionate influence of outliers on the variance. The method developed in this dissertation effectively avoids this problem, and is more efficient than the Maximum Variance criterion. It can be applied for any regression. A detailed demonstration will be presented in Chapter 7, in which we will show how the Maximum Inter-Quartile Range (MIQR) criterion avoids the "dead valley" problem while Maximum Variance wastes training.

Maximum Error criteria are derived from IMSE (Integrated Mean Squared Error) [88]. Much previous work in active learning addresses a special case of the IMSE criterion

that arises by making the strong assumption that functions to be modelled should be reasonably smooth. In this case, exemplars are selected to minimise model bias. Besides, methods derived from this criterion often require deterministic target information or labelled data. Although this approximation is valid asymptotically as the amount of training data grows large, it can be quite poor for functions which are not reasonably smooth. In Chapter 7 it will be demonstrated that our method works well even when it approaches the square wave, a typical discontinuous or not reasonably smooth function. In addition, we present the results of benchmark studies with the Maximum Inter-Quartile Range technique developed in Chapter 6. We intentionally introduce two distinct functions: one is a continuous function, cos(x), and the another is a discontinuous function, the square wave. Experimental results using active learning and passive learning are discussed in detail. The experiments confirm the advantages of the new method developed for training data selection over traditional passive learning. Intuitively, training is more difficult for a discontinuous function than for a continuous function. Thus the training set and network complexity for approximating a discontinuous function are bigger than those for approximating a continuous function. It has been found that the resource demands of approaching a discontinuous function are significantly more than those for a continuous function, no matter whether active learning or passive learning is used. It is also found that some data points are particularly useful and the order of exemplar selection is very interesting when approximating a discontinuous function. Since our method has successful application for both continuous functions and discontinuous functions, it contributes a significant advance for a wide class of learning tasks which cannot be tackled by other existing methods.

In Chapter 8 more details of the sensitivity analysis of neural network ensemble learning are presented. It is concluded that growing the ensemble of networks can improve data selection, generalisation performance, network learning efficiency and resampling efficiency. However, this improvement is restricted by linearly independent networks of the ensemble. The experimental results show that the ensemble consisting of 15 networks is sufficient to approach any function with a good generalisation. This fact is consistent with theoretical result[45][17]. It is also indicated that growing the ensemble of networks does not influence the network complexity.

Finally, in Chapter 9 conclusions and future work are presented. The power of the new method (MIQR) lies in combining the neural network ensemble and nonparametric statistical techniques (inter-quartile range) to implement "the minority obeys the majority" philosophy. MIQR's principle and essence is to embody "fairness" by increasing the size of the ensemble, reflecting the majority's desire without considering the minority (outliers). The new method's contribution and importance is that it not only plays a role in active learning, but also it outperforms current contending methods (Maximum Variance and Maximum Error), perhaps becoming the most effective active learning technique. In particular, the ideas behind MIQR can be applied to other scientific research, educational methodology and societal behaviours.

Future work should consider several respects. First the environmental distribution of different data points should be investigated. Second, what if several examples could be resampled simultaneously at a time instead of only one? Third, now that MIQR's power lies in "the minority obeys the majority" philosophy, thus the variation of the ratio of the minority (outliers) to the majority (inter-quartile range) must influence with the results of data selection. Further investigation with this would be interesting and necessary.

Chapter 2

Active Learning and Its Development

2.1 Introduction

This chapter provides a review of subject matter addressed in this dissertation, which states the essence of the philosophical ideas behind it.

The idea of *active learning* originates from findings in cognitive science or selective attention. In the case when a student or a *learner* tries to learn some new concept, progress often takes the form of an iterative *bidirectional* information exchange: not just being a *passive* recipient of various descriptions and facts concerning the *problem domain* given by a teacher (oracle), the learner, on the other hand, can make intelligent queries, focusing on interesting or confusing facts, while gaining more understanding of the underlying concept.

In recent years, this active learning phenomenon has attracted considerable interest among the neural network research community. In this context, the *learner* is a neural network model parameterised by a set of connection weights, and the *problem domain* is characterised by a collection of limited or unlimited data samples or exploratory space, with the data drawn from applications such as function approximation[89][99], pattern classification [112], time-series prediction [81] and robot path exploration [22] [100] etc.

The problem of "active learning" has been extensively studied in economic theory and statistics [32][34]. Lindley and Luttrell used experimental design methods as an objective function within a Bayesian framework [65][68]. David MacKay used a similar information-based objective function and discussed the problem of optimal data selection for neural network training[70][71][72]. MacKay demonstrated that learning can be made more efficient if we can actively select particular salient data points. Within a Bayesian learning framework, he establish three criteria for data selection [73]. All these criteria, however, depend on the assumption that the hypothesis space is correct, which, he admitted, may prove to be their main weakness. Besides, some early connectionist approaches toward active learning include: Ahmad and Omohundro[1] on training networks by selective attention; Eberhart [29] and Hwang et. al.[51] on a query-based neural network learning scheme that generates queries near classification boundaries; Plutowski and White[86] on an efficient feedforward network training technique that selects new training examples with maximum potential utility from among available candidate examples. Both Hwang et.al and Plutowski et. al. choose new training examples according to information derived from a partially trained network.

Some major benefits and concerns about active learning, sometimes called *Query-based learning* or *Active data selection*, are as follows: first, in *passive learning* the input to a neural network is sampled at random according to some probability distribution and the target at the output is then provided by a teacher from a problem domain. Random examples contain less and less *new* information as learning proceeds. By concentrating on learning those most informative or "maximally useful" examples [96], we expect to gain better generalisation performance. Secondly, there often exists the case where *data collection* is very expensive, or given an input vector, labelling its target value is costly. This is the situation in chemical processes and precision instrumentation experiments. In such cases we wish to conduct the minimum number of tests to generate the most informative data in order to model the underlying system mechanism correctly.

2.2 Active Learning

In the neural network community, active learning encompasses two distinct techniques for choosing training examples, namely "active sampling" and "active selection". The former's task is to add new examples to the set of available examples, *i.e.* search for new information, and latter is to select training exemplars from the existing set of available examples, *i.e.* sift or filter particular information. We will discuss in more details as follows.

2.2.1 Active Sampling - Searching for New Training Examples

Active sampling is essentially to search for new training examples, being concerned with determining the distribution of training examples, which is referred to as the "sampling distribution". In traditional approaches, training examples are sampled evenly or randomly from the input space, which is known as the "natural distribution" or the "environmental distribution". Due to a fact that active sampling is a search for particular training examples, active sampling may result in a modified sampling distribution which is different from the "environmental distribution" according to which examples occur naturally in the learning environment[88].

Judicious sampling utilises prior knowledge to determine the sampling distribution. This prior knowledge includes the model (neural network architecture), the learning algorithm, and the learning task including the environmental distribution and any prior knowledge of the target function. Given enough knowledge about the learning task and the network model, it is possible to determine the optimal sampling distribution in advance, without need for active sampling [3][34].

Active sampling provides the means for querying the environment for new information, for instance, to fill in the gaps in a finite sample due to random deviation in the location of examples, or to verify particular information. Active sampling exploits knowledge gleaned from examples provided so far to determine regions of uncertainty. Active sampling

determines the sampling distribution according to the result of learning based on previous examples and therefore, in this sense, the network itself actively participates in its own training procedure.

Typically, the task is to minimise training examples, or to confirm some particular information crucial to the estimate. If data measurements are costly or there is uncertainty in the data distribution, active sampling is very necessary and desirable. Historically, active sampling has been referred to variously as "active learning", "sequential design", or "query-based learning". In a loose sense, we can think of a concise set of exemplars as analogous to the notion of "extreme points" employed by Cover for classification tasks[26]. These extreme points are so distinguished from other points that they play a key role in function approximations (details will be demonstrated in Chapter 7). Active sampling is often called query-based learning as it "queries" an "oracle" for the value of the function in these regions [20][21][111] [51].

The remainder of this section contains a critical review of active sampling. This will include reviewing a novel stochastic/random sampling method and applications where knowledge of either the learning task or the model is exploited. Finally, we will review methods which minimise model variance.

2.2.1.1 Learning Task Generalisation and Its Algorithm

Query-based deduction programs are studied in the computational learning theory literature [105] [38][49][61]. Hellerstein and Karpinski study the exact learning of a class of Boolean tasks. They use a combinatorial characterisation of the learning task to prove that learning by queries to an oracle permits feasible time algorithms for this class of problems, whereas learning these tasks from examples drawn at random is not possible in polynomial time[49]. In some formal situations, active sampling is provably more powerful (in computation theoretic terms) than passive inductive inference[38], which demonstrates that learning from responses to queries provided by an oracle extends the set of functions which can be inferred by inductive inference.

This work provides theoretical justification for the use of active sampling for reducing the computation required by learning, or for extending the class of problems which can be learned. However, none of the results provides useful algorithms. Baum proposed an active sampling algorithm for iteratively seeking out classification boundaries for which a convergence proof is supplied[8]. Plutowski does propose an effective and practical algorithm for both clean and noisy data selection[88]. However, all of these results apply to either Boolean tasks or labelled data, or reasonably smooth functions, whereas we are interested in the more general task of learning arbitrary mappings. Although RayChaudhuri[89] and Krogh[59] also introduce ensemble networks so that their methods are applicable to unlabelled data, they do not present an effective and practical algorithm because their methods are not able to automatically adapt network complexity and the training threshold. Besides, they use Maximum Variance methods for data selection, which is found to have lower efficiency in approach a more complicated mapping such as a sharp discontinuous function (this will be demonstrated in Chapter 8).

2.2.1.2 Prior Knowledge for Active Sampling

Sollich provides theoretical results which indicate that active sampling can give a significant benefit[97]. He uses Bayesian theory, together with the assumption that prior knowledge of the learning task is available. Prior knowledge is required of the prior distribution of the target functions, and also, for some of the results, of the correct model complexity required to parameterise the target function. This allows analysis of the benefit of a particular set of exemplars, as the formulation allows one to formally state the probability of being provided with particular set of exemplars by the "teacher".

Results are obtained for three particular applications. the first being a simple (although nonlinear) classification task, the second being the task of training a perceptron, and the third, a task where model complexity is not known in advance.

First, we discuss the results for a linear perceptron. In the limit of an infinite number of training examples, although active sampling does no better than random sampling at minimising generalisation error, it does provide a significant benefit when training on finite size training sets. Moreover, Plutowski confirms that the improvement in generalisation due to active sampling depends greatly upon the ratio of exemplars to the number of weights. The improvement due to active sampling increased dramatically as this ratio rose from 0 to 1, peaking shortly thereafter, and then more slowly decreasing as the ratio increased.

Now we turn to the results for the simple nonlinear task, where parameterisation of the target is known in advance. Sollich points out the important fact that active sampling algorithms can be derived according to two general classes of objective criterion, one maximising information gain, and other minimising generalisation error. It is demonstrated that an active sampling algorithm designed to minimise the information-theoretic difference between student and teacher will not, in general, perform as well (in terms of reducing generalisation error) as an algorithm designed to minimise generalisation error[107]. However, both approaches cause generalisation error to decrease exponentially[74].

Finally, Sollich considers the case where the network model complexity is not optimal. In this case, it is possible for an active sampling algorithm to be detrimental. This can occur when training examples are very noisy, and the number of exemplars is small. In these cases, the active sampling algorithm can be overconfident in its selection, which can ultimately lead learning astray in the presence of noise. However, in all other cases, active sampling algorithms will give better generalisation in the limit (as the number of exemplars grows large), as well as for finite sized training sets, in which case the relative improvement in generalisation due to a single judicious query is, in general, greater than that due to a single additional random example. These results give strong support for the development of active sampling methods.

One approach to sampling is to sample randomly at first, until either the environmental distribution of examples is well-estimated, or the form of the desired function is becoming apparent, and sample actively thereafter. In this approach, the sampling distribution is dynamically determined from information acquired through prior training.

One example of this is the stochastic or random sampling approach [111]. Somewhat related is [51], where knowledge of the model is exploited after likewise training on an initial set of examples obtained "passively". RayChaudhuri [89] and Raviv [90] use this approach in their experiments. This provides an active sampling technique for querying the environment both for clean data and in the presence of noise. It does require preparation by training the model upon a small sample obtained "passively" or "randomly" before beginning with active sampling. This approach is used in all our experiments. We have found this approach to be effective in practice.

Plutowski provides a convergence result[88]. The proof requires an ergodicity assumption, implying that the sampling distribution is asymptotically dense. In practice, one approach is using the bootstrapping technique [9][30][31], which has been demonstrated by a number of researchers[80][89][90][82].

2.2.1.3 Using Task Knowledge of the Application

A common method in active sampling is to use a partially trained network to determine *regions of uncertainty* in the environment and then query the "oracle" for values of the function in these regions. The differences among the methods lie in how "uncertainty" is quantified. Here we present three approaches where task knowledge is exploited to optimise the sampling distribution.

Text Classification

Lewis and David [64] define text classification as the automated grouping of textual entities. In this application, text is cheap, but the process of determining what classes an example of text belongs to is expensive. A large number of "unlabeled" examples are

generated, where no target information is made available. A subset of these are presented to an oracle for labelling. The algorithm was able to reduce the amount of data that needs to be classified. This article also provides a comprehensive review of other work done by applying active sampling to this learning task.

Visual Learning Task

In visual learning tasks an intuitive heuristic technique is used to exploit concepts of visual attention inspired by cognitive modelling [1]. During training, the system needs to search for the most relevant portion of the image, and then acquires information from the selected portions. This approach gives very efficient algorithms for solving certain learning tasks involving clustering analysis. Although rapid adaptation is available using a local searching technique in receptive fields, the task performed is essentially a toy problem because the learning task is well-defined in advance, as is the network model. The method of selecting data points exploits regularities in the learning task as well as in the network model to obtain data which is intuitively most informative to the network learning rule. So far it has not provided powerful evidence of how to extend this intuitive heuristic technique for practical application learning.

Yamada and Cottrell have developed this approach particularly for the purpose of face recognition[110]. Yamada uses Gabor transformations on image data to produce features which can be localised to particular regions of the image. Then, a heuristic chooses those features which are most informative to the recognition algorithm. The result is a face recognition algorithm which operates without needing to process all of the available data, by processing selected subregions of the image. However active sampling only plays a minor role in this approach because much of the sampling distribution is determined in advance.

Robotic Exploration

In robotic exploration, Thrun and Moller's work may be typical. Thrun and Moller present an adaptive strategy for efficient exploration in non-discrete dynamical environments

[100]. In their experiments, they used a so-called competence map to train the world model and estimate the competence (error) of it. In training the competence map was used for driving the "learner" to less familiar regions. In order to avoid unnecessary exploration costs, a selective attention mechanism is used to switch between exploration and exploitation. The resulting learning system is dynamical in the sense that whenever one particular region in state space is scanned for many runs, sooner or later the exploration mechanics forces the learner to leave this region. This exploration technique has been demonstrated as being useful on a robot navigation task. However, the exploration method they presented would need to explore the whole state-action space. This may be unreasonable for a very large exploration space. In order to deal with large exploration spaces, this method should be extended by some mechanism for cutting off exploration in "irrelevant" regions in the state-action space, which may be determined by some notion of "relevance".

Cohn[22] also considered the problem of learning input/output mappings through exploration, e.g. learning the kinematics or dynamics of a robotic manipulator. If actions are expensive and computation is cheap, then we may explore by selecting a trajectory through the input space which provides the most information in the fewest number of steps. He uses results from the field of optimal experiment design to guide such exploration, and demonstrate its use on a simple kinematics problem.

Determining Sampling Space

One problem common among sampling techniques is that the cost of computing optimality over the input space scales exponentially with the dimension of the input space. In order to reduce the cost in question, an alternative is to selectively sample regions of the input space. Hwang, et.al propose a concept of "boundary" and point out that neural network learning of classification tasks is to generate exemplars in the vicinity of classification boundaries [51][52]. This technique is intended for use on a partially trained "classifier"(network) for learning a discrimination task. This is often referred to as "classification" in the neural network literature. They also develop a useful sampling

technique by (implicitly) assuming a uniform distribution over the input space, using a heuristic that determined the classification boundary of the network by inverting the network function. It is assumed that boundaries are regions where additional information would most directly test the network fit [51]. According to the theory, examples are selected from the set of inputs in the inversion of the network function. Intuitively, these examples correspond to regions of maximum classification ambiguity. Although intuitively pleasing, this heuristic does not admit a general technique for learning arbitrary mappings because no explicit objective function is provided and optimised. It is also not clear how well it would perform with an arbitrary distribution of examples in input space. The heuristic relies heavily upon the structure of the learning task. Examples selected accordingly are not necessarily informative, because it is not clear how well it would perform at locating new examples.

Today the technique for determining the sampling space is improved. It is such maximum "ambiguity" that Krogh and RayChaudhuri use in their experiments although the input space is one dimension [59][89]. Actually, the technique developed in this dissertation also follows the same theory as Hwang, et.al, but has been improved. In practice, the networks are well trained first, then the sensitivity of the output with respect to each input is done through computing the variance as Krogh and RayChaudhuri or through computing the inter-quartile range as developed in this dissertation. Finally the technique chooses input examples for which a slight perturbation of the example causes a large perturbation (here, maximum variance or maximum inter-quartile range) in the network output.

2.2.1.4 Minimal Model Variance Techniques

Active sampling techniques are discussed earliest and most extensively in the statistics and econometrics literature, such as in the theory of *response surface methodology, optimal experiments* and *sequential design* [34][77][105][12][57]. The general task is to estimate an arbitrary, unknown mapping in the statistics and econometrics. These approaches are typically Maximum Variance sampling criteria, utilising the Integrated Mean Squared

Error[88] for the purpose of sampling the environment for new training examples[3][25][77][76][94]. Amongst these researchers, Fedorov gives the most complete treatment, and obtains several active sampling methods for performing interpolation and extrapolation from noisy real valued data, including a method for obtaining new data which is maximally informative with respect to comparing competing models. These approaches require a deterministic or correctly specified model, i.e., these approaches require a rather strong assumption that given sufficient training, the model bias (approximation error) can be ignored, resulting in techniques primarily devoted to minimising some measure of model variance (estimation error).

These results have been extended to be applicable for nonlinear models. However, the extensions are subject to stringent conditions so as to assure that a linearised version of the model can be obtained to allow the application of the techniques, which are developed for linear models. These stringent conditions restrict their practical application because when the function to be modelled is not reasonably smooth, we cannot assure that linearisation of the model yields a good approximation. Plutowski used these minimum-variance techniques to prune away examples from noisy data after training is finished. In fact, using his Δ ISB criterion, the method for selecting training examples is directly derived from IMSE, therefore it is not effective and efficient for discontinuous functions while it is a successful technique for reasonably smooth functions.

2.2.2 Active Selection - Sifting or Filtering Particular Information

Active selection starts with a given set of examples, and selects a subset to use as training "exemplars", exploiting information obtained from the model due to learning from previous exemplars. It relies upon the environmental distribution to generate a set of candidate examples large enough to ensure proper coverage of the example space. Active selection is used to deal with the redundancy of an oversampled set of examples to reduce the size of the training set. Previous work on active selection has been done by many researchers [97][88][59][112][73][89].

2.2.2.1 The Version Space Technique

Cohn examines a general framework for learning Boolean mappings[20][21]. The task is to correctly "classify" new points drawn from a space X according to a distribution u. When learning from examples drawn according to input distribution p, the probability that a new example allows a reduction in error is exactly the probability that the current hypothesis will misclassify the example. Obviously we would want to select points which can provide useful information with high probability. Specifically, we may choose examples lying within a known region of uncertainty. Such examples are those for which there exist two different hypotheses (in the current hypothesis space) which are consistent with all previous examples, but which differ on the new example. In network terms, this is equivalent to having two networks which classify all previously encountered examples correctly, but which differ on a new example. An example drawn according to p would be presented to the oracle for classification only if it has this property.

This is referred to as a *version space technique* [20]. Cohn points out that the version space of models that are consistent with the given set of data is reduced with each new example. This approach assumes that the acquisition of an example occurs in two stages. An input location is obtained by a random draw on X according to p. Then, the location is presented to an oracle for classification. The algorithm estimates the worth of a particular example relative to other available examples, therefore, it is a way of selecting exemplars from a set of available examples. Furthermore, the model does not modify the sampling distribution since the oracle only accepts inputs according to the environmental distribution, p. Therefore, this can be viewed as an exemplar selection technique.

While this work provides useful algorithms, they are, in general, computationally intensive, and they are intended for Boolean mappings, whereas we are interested in the more general case.

2.2.2.2 Minimum Variance Pruning

Note that the methods mentioned above for pruning away examples after training are essentially active selection methods, since they are a way to select a subset from a set of available examples [34][76][94][71][73]. This technique can be used to evaluate the importance of exemplars after training is completed. A certain subset of the training examples may be critical to the form of the final estimate obtained from the initial set, in that a small amount of noise applied to these examples can have significant impact on the network estimate [34][76][94]. In this sense, the approach is concerned with determining the sensitivity of the network estimate to particular training examples. Given knowledge of this sensitivity, we may either resample particular examples, or prune certain examples from the training set. However, although Plutowski effectively implements active pruning away of exemplars which may still contain "outlier" information, his main drawback is to use only a single network, therefore, he additionally increases computational costs because of computing the Hessian matrix at each iteration.

2.2.2.3 Nonparametric Applications

While much of the work above has been applied to nonlinear models, it was developed largely for linear models, or to discrimination learning tasks. If it is known that the function to be learned is reasonably smooth, a criterion exists for choosing examples for a nonparametric, nonlinear, kernel-based smoothing approach [33][42]. However, a large amount of data is necessary to regulate model complexity.

Faraway uses a kernel-based method to achieve nonparametric estimation of a continuous mapping [33]. This is a localised approximation, where each kernel gives an essentially local contribution to the global mapping. Previous work indicates that if the kernels all use the same global "bandwidth"[42] then asymptotically the optimal distribution of examples is evenly spaced [33]. Furthermore, if some degree of local smoothing is applied to the data, then the optimal distribution of examples (in input space) is a roughly approximated function of the target function g [33].

Faraway investigates the possible benefit of actively sampling training examples for the purpose of training a kernel estimator using locally variable kernel bandwidths, that is, where the size of an individual basis function can vary. Faraway concludes that local smoothing must be used so that actively selected data outperform the evenly spaced distribution when globally constant bandwidth kernels are used. Hence Faraway focuses on deriving an active selection technique for training a kernel regression model with variable kernel bandwidths, using a data-adaptive method for selecting the local kernel bandwidths. A crucial element of Faraway's approach is the selection of the global bandwidth, as a starting point for selecting good local bandwidths. In Plutowski's work, selection of the global bandwidth is analogous to selecting the network complexity. He firstly uses an initial sample of size n to select a good global kernel bandwidth, which is done by choosing the global bandwidth minimising IMSE over the initial sample. Then he uses an estimate of the second derivative of the target function g to obtain the local bandwidths for each kernel, whose calculation utilises an asymptotic expansion of IMSE.

The optimal sampling distribution is known to be proportional to the second derivative of g [88]. By comparing the active sampling distribution with the optimal asymptotic distribution, Faraway concludes that variable kernel bandwidths must be used for this technique to have an advantage over an evenly spaced distribution of examples. Plutowski also arrives at the same conclusion. Moreover, he also confirms that this technique is definitely advantageous over randomly sampling.

We note that the estimate of the local bandwidth requires estimation of the second derivative of g, therefore, g must be twice continuously differentiable. However, many tasks involve discontinuous functions as we will see in Chapter 7. Thus Faraway and Plutowski's theory, which is based on the assumption that the function to be learned is reasonably smooth, would not be applicable to these problems. Our work is inspired by Faraway and Plutowski, but highly differentiated from theirs. First, we use a set of ensemble networks, thus, selection of the global bandwidth is analogous to selecting and combining a set of different networks. Secondly, our technique will be applicable to

general cases rather than only to a class of "reasonably smooth functions". Thirdly, we simplify active selection since our algorithms need not compute the Hessian matrix of g. Finally, we will demonstrate that our method is more effective and efficient than those described in previous work.

Once a set of networks is combined and estimated, to calculate the local bandwidth requires evaluation of the estimate of the inter-quartile range over all available examples. Also, the new examples must be selected from a set of available candidates, and so, in our terminology, this an active selection technique.

2.3 Active Learning in Practice

Having stated the objectives that active learning is meant to achieve, we need to look at the principal ways in which this idea is developed in practice. Major efforts are centred around formulating effective *learning frameworks* and/or introducing *optimal criteria* whereby intelligent data selection can be made. Active learning involves two tasks, namely, data modelling and data selection.

Starting with a small random data set (sometimes a single datum), the *data modelling* process fits the data examples seen so far to a neural network model using an optimisation method [91]. Care needs to be taken not to over-fit the model, and in some cases the model selection process is based on cross-validation techniques [98][10][59] to avoid this problem.

Based on the model(s) thus obtained, next in the *data selection* process, we define and compute an *optimisation criterion*, for example the *variance of output estimates* [23], which corresponds to the model's estimate of the expected squared distance between its *actual* output and the as yet unknown *true* output for a *new* data point. We then choose to add to the training set the data point that minimises this variance. Note that this variance is

a simplified measure of *integrated prediction error* [87], and different approaches have different ways modifying it, for example, in this thesis a new optimisation criterion, the *inter-quartile range of output estimates* will be proposed and discussed in detail in Chapter 6, Chapter 7 and Chapter 8.

Some important features regarding the current techniques for active learning are summarised below.

2.4 The Learning Framework

In most studies seen in the literature, researchers tend to advocate the framework of so-called *optimal experiment design* [4] popularised in statistics, which draws heavily on the technique of Maximum Likelihood Estimation (MLE). The work carried out in [72] by MacKay, and by Cohn[22] and his co-authors [23] all follow this line of thought, though for computational efficiency, the objective function, *integrated prediction error*, for selecting the new data point needs to be approximated by an asymptotic normal approximation [87] or expansion. In one case, a Markov Chain Monte Carlo simulation was introduced [56] to approximate this measure. Note that these methods fall in with the Bayesian inference paradigm [97].

Yet another framework to be considered is *sequential optimal recovery* [79]. This type of analysis concentrates on the worst case behaviour of the algorithm and no probabilistic assumptions are made. It has been subjected to a detailed study in a scenario of function approximation, in which one tends to compute the maximum possible error for guidance in selecting the next data point.

2.5 Query Construction, Filtering and Data Subset Selection

In active learning we may need to distinguish between three different situations as regards the new data to be explored. Assuming that a neural network model has been established based on examples (*input-output pairs*) seen so far, and the thorough exploration of data space would include an incremental expansion of the training set: if the *input* value of a new example is unknown, for instance, in such a dynamical environment as robotic exploration, we need to calculate the input value (maybe stochastically) to be queried. This is called *query construction* [97]; If there exists a string of random input values that can be queried, this is normally referred to as *query filtering* [87]. In traditional systems that learn from examples, random examples contain less and less new information as learning proceeds. Therefore, generalisation performance can be improved by learning through "filtering" data, *i.e.*, by deliberately choosing the input of each new training example. Note that these new data may be labelled or unlabelled, that is to say, within *input-output pairs*, the outputs may be known or unknown corresponding to the inputs. If, on the other hand, the data we are to deal with are labelled and of fixed size, the process of adding more examples to the training set is often called *data subset selection*; see, for instance, the work by Zhang [112] and a similar strategy suggested by Cachin [18].

2.6 Query by Committee

Query by committee has been investigated by a number of researchers [62][95][35][75][60]. When, in the data modelling process, a "committee" of neural network models is used, the *disagreement* in predicted values between members of the committee, for arbitrary input data, would provide an appropriate guide about where to sample the new data in the problem domain. The disagreement can be expressed as a measure of *ambiguity* as in [59], or as the estimated *output variances* as in [89] and in [56] when applying these models to a sequence of unlabelled data. The ways of acquiring these committee models are different in each case. Of most interest is the work by

Kindermann et al. who used bootstrapping techniques [31] which have found wide applications in time-series predictions for achieving better generalisation performance and accurate estimation of output error bars [14] [101] [102]etc. Note also that Freund et al. [36] have derived some interesting results, for a two-member committee, regarding the relationship between the prediction error and number of queries required.

2.7 Relation to the State of the Art - Differences from Previous Work

The approach which is the subject of this dissertation is similar to many previous approaches, in that examples are chosen according to information derived from partially trained networks. In this respect, our methods are close to Faraway and Plutowski's technique. Using a formulation applicable to general purpose nonlinear regression, we adapt techniques for the purpose of approximating all functional space, including discontinuous functions, which breaks the assumption that the target function g must be reasonably smooth as it is proposed by most researchers who use Maximum Error criteria. Furthermore, we will not use a single network to solve mappings, but instead we use a set of ensemble networks. Thus our approach is distinguished from previous work that is derived from a Bayesian framework[79][99][87][23][33][52][73]. In this sense, our methods are closest to RayChaudhuri's technique, but different in terms of the criterion for selecting training examples. We propose the Maximum Inter-Quartile Range criterion (MIQR) directly derived from nonparametric statistics [27]. According to this new method, exemplar selection is not unduly influenced by *"outliers"*, rather, is principally dependent upon the *"mainstream"* output of the ensemble networks, i.e. "the *minority obeys the majority*". The MIQR criterion, in particular, when it is applied to discontinuous functions, outperforms the Maximum Variance criterion, which is that most used by previous researchers [59][89] etc and is more effective and efficient in practice. This will be demonstrated in Chapter 6, 7 and 8.

Although Plutowski's approach is also nonparametric in the sense that the complexity of the network need not be pre-specified, improving generalisation performance relies only upon judiciously sampling techniques. Our approach for improving generalisation not only includes sampling techniques, but also includes ensemble methods, which not only simplifies the algorithm and reduces computational complexity considerably, but also enhances flexibility. In particular, our work shows that by using ensemble networks, we can considerably reduce network complexity as well, with only one hidden-layer network architecture with a small number of hidden units.

Although we focus our methods on clean data or a noiseless environment, it would be easy to extend our methods to a noisy environment, since our methods for selecting training examples is orientated to unlabelled data selection. In fact, RayChaudhuri has done similar work. He demonstrated that active learning techniques outweigh passive or random data selection even if in the presence of noise [89].

Reviewing previous active learning methods, such as Maximum Error and Maximum Variance, it is found that Maximum Error not only is computationally complicated and expensive, but also requires the assumption that the function to be modelled is reasonably smooth because these Maximum Error methods have to compute the inversion of a huge Hessian matrix such as Plutowski's Δ ISB and Sung et.al's EISD(expected integrated squared difference). It is unable to approach a discontinuous function such as the square wave, because we can not compute the Hessian matrix. In addition, Maximum Error only uses a single network, therefore it cannot make use of the advantages of ensemble networks. Furthermore, Maximum Error only approaches labelled data, lacking a clear stopping criterion. Maximum Variance methods are very popularly used in ensemble network learning. However, in practice, it is found that the Maximum Variance method frequently traps into a local minimum and quits training too early or fails to obtain a desired training level. Theoretically, exemplar selection by the Maximum Variance method is essentially based on the assumption that values used in the computation of variance

should be normally-distributed. The frequent presence of outliers in network outputs indicates that this is not the case. As a result, when using the Maximum Variance criterion, the training may be trapped in a "dead valley" and wastes a large number of useless iterations because of the disproportionate influence of outliers on the variance.

Along the way, our work touches on the issue of exemplar selection and generalisation, justifying the use of ensemble networks and methods for selecting training examples, and relating these measures to other measures of the active sampling technique and generalisation more commonly referred to in the neural network literature. Furthermore, our current work in obtaining a method for selecting exemplars by using an ensemble network raises the question of what it means for an exemplar from ensemble outputs to be an "outlier" in a small set of ensemble networks, a concept addressed in more depth in the study of robust statistics. More work is required to evaluate the strength of the connection between our work and the results of robust statistics. Finally we conjecture the possibility of simultaneously choosing several points when using our criterion for selecting training examples since our learning algorithm is backpropagation (based on the least squared error), and therefore there exist several local maxima or minima. If our conjecture is justified, we can speed up the training procedure and increase network learning efficiency.

Chapter 3

A Brief Introduction to Neural Networks

3.1 What Is A Neural Network ?

A neural network is an information processing system. In contrast to traditional information processing systems such as computers or AI, it is nonalgorithmic, nondigital, and intensely parallel. A neural network is composed of a number of very simple and highly interconnected neurodes, which are the analogues of the biological neural cells in the brain. In a neural network, signals are transmitted or passed over these neurodes, which are connected by a large number of weighted links. Typically, each neurode receives many signals over its incoming connections; some of these incoming signals may be generated by other neurodes, and others may come from the outside world — an input signal pattern presented to the neural network is designed especially for a particular problem. Usually each neurode has many of these incoming signal connections; however, it never produces more than a single outgoing signal. That output signal transmits over the neurode's outgoing connection, which usually splits into a very large number of smaller connections, which terminate at a different destinations. However, each of these branches of the single outgoing connection transmits the same signal. Most of these outgoing branches terminate at the incoming connection of some other neurode in the neural network; others may terminate outside the neural network (the final outputs) and generate control or response patterns. Figure 3.1-1 illustrates the physical connections of a typical

neural network. These fundamental connection rules which are used in artificial neural networks are directly inspired by the architecture of the human brain. Thus, neural network researchers have been studying the biology of the human brain in an attempt to set up successful neural network architectures by imitating human brain architectures. Of course, the artificial neural network's neurodes are not the same as their biological roots, and only a very rough approximation of a biological human brain neuron. As a result, current artificial neural networks cannot be thought to function in exactly the same way as a biological neural network does.

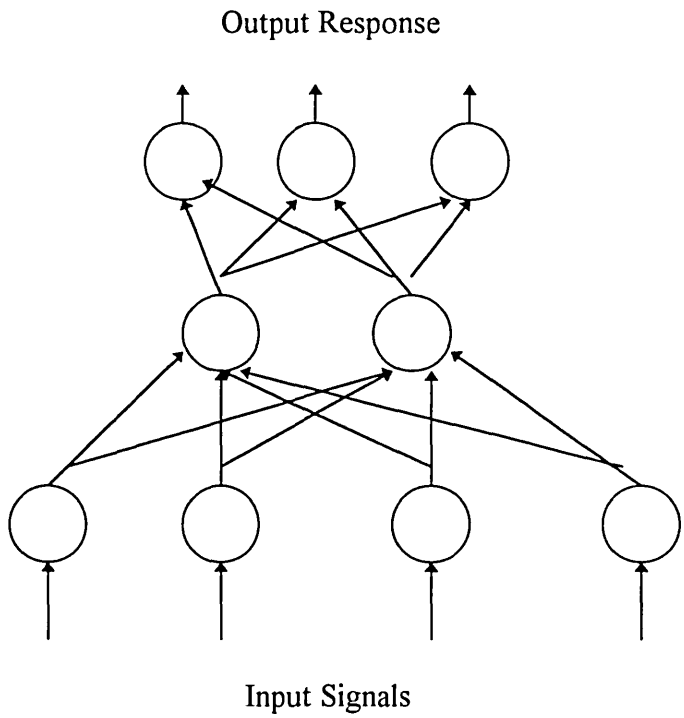


Figure 3.1-1: Typical Neural Network Architecture

A neurode in a neural network is an extremely simple unit. It receives input stimuli (or signals) along its input connections and **translates** those stimuli into an output response, then the output response is transmitted along the neurode's output connection. The place where the input connection meets the neurode is usually called the *synapse*, which is taken from the biological neuron's synapse. The mathematical expression that describes the

translation of input stimulus pattern to output response signal is called the *transfer function* of the neurode. The translation operation is carried out by following three steps.

First, the neurode computes the weighted inputs received along its input connections. This can be done in several ways, but most commonly consists of computing the value I_i as shown here:

$$I_i = \sum_{j=1}^{n} w_{ij} x_j$$

In this expression, I_i is the net weighted input received by neurode i from a total of n neurodes in the network. The incoming signal from the jth neurode is designated by x_j, and the weight on the connection directed from neurode j to neurode i is designated by w_{ij}. It may be thought that strong input signals will be significant and weak will be lost.

However, a strong input signal, if it is arriving over a weakly weighted connection, may have less effect than a weaker signal arriving over a strongly weighted connection. Thus, the net stimulus I_i is the collective contribution of all arriving signals, not the value of any particular one. A weight may be negative instead of positive. In this case the connection is said to be inhibitory; that is, it tends to reduce the overall stimulation of the receiving neurode.

The second step of the translation operation represented by the neurode's transfer function consists of converting the net input to an activation level for the neurode. The activation level of the neurode is equivalent to the level of excitement of a biological neuron. In most neural networks, the activation function is a sigmoid function; that is, the activation is expressed by an S-shaped curve. The most commonly used sigmoid function is:

$$f(I) = \frac{1}{1+e^{-I}}$$

This function has the useful property that its derivative is exceptionally easy to compute:

$$\frac{df(I)}{dI} = f(I)\,(1-f(I))$$

Not all neural networks use such a sigmoid function, but the most effective networks usually use the function. The exact form of the sigmoid function is not particularly important; it is merely important that the function be monotonically increasing and bounded with both lower and upper limits.

The final step accomplished by the transfer function is to convert the neurode's activation level to an output signal. Most commonly, this is done by setting the output signal to the following expression:

$$y_i = \begin{cases} f(I), \text{ if } f(I) > T \\ 0, \text{ otherwise} \end{cases}$$

where T is a threshold value. In other words, the neurode's output is its activation level as long as that activation value exceeds a given threshold; otherwise the neurode does not output anything.

The transfer function used by the neurodes in a particular network is nearly always the same for all neurodes in the network or at least for all neurodes in a significant subsection of the network. The exact form of the transfer function can be modified from those suggested above. For example, the sigmoid function used can be quite different from the one proposed. Some researchers prefer to use the hyperbolic tangent function because it has an easily obtained inverse and is symmetric around zero. It is also possible to use other variations of the function.

3.2 A Neural Network in Operation

When a neural network, for example, the one in figure 2.1, is presented with an input pattern, each neurode in the input layer of the network receives a small piece of the input pattern. Usually these neurodes distribute their small parts of the pattern as "fan-out" signals to the neurodes in the middle layer. Therefore each of the middle-layer neurodes in

general receives the entire input pattern. However, the pattern has been modified over the weighted connections from the input layer to the middle layer. So, each of these neurodes in the middle layer has a somewhat different version of the input pattern from its neighbours because the weights on the connections are usually different for each middle-layer neurode. Accordingly, some combination of middle-layer neurodes will become active to varying degrees, depending on the exact collection of weighted connections from the input layer to middle layer. Thus a variety of output responses results from the middle-layer neurodes, some may produce no output at all, some possibly producing an extremely strong output.

In most cases, all the middle-layer neurodes will transmit their output signals to all of the neurodes in the output layer; thus, each of the output-layer neurodes respectively receives the complete pattern of output activity from the middle-layer neurodes through weighted connections. However, just as took place with the middle-layer neurodes, this activity pattern is modified by passage through the weighted connections between the middle- and output-layer neurodes. So again each neurode in the output layer has a somewhat different version of the middle-layer activity from its neighbours. As with the middle layer, the result is that some combination of output-layer neurodes becomes active to greater or lesser degrees, causing them to output a signal. This pattern of output-layer neurode responses is the network's final overall response to the original input pattern stimulus.

Obviously the design of a neural network is not limited to a three layered architecture. Although the three-layered network is the most commonly used form, networks may have as few as one or two layers, and as many as a dozen or more layers. In all cases, however, the process carried out by the networks is the same: they receive an input stimulus pattern from the input layer of the network, transmit signals through the weighted connections between layers and modify the resulting patterns by transfer functions, and finally output in the output layer of the network.

3.3 Learning and Training in A Neural Network

Neural networks can solve many problems. Unlike traditional computing methods, they solve problems by ***learning*** instead of by ***programming***. Learning and training are the most important characteristics in a neural network's process, and therefore are fundamental to nearly all neural networks. Learning is achieved not by modifying the neurodes in the network but by modifying the weights on the interconnections in the network. Normally (although not always), the transfer function is assumed fixed, thus each neurode's output is determined by two things only: the incoming signal and the weights on the input connections to the neurode. Clearly if the neurode is to learn to respond correctly to a given incoming signal pattern, the only possible element that can be used to improve the neurode's performance is the weight on the connection.

Learning has a close relationship with training; learning is achieved by training. But training and learning are not the same. Training is the procedure by which the network learns; learning is the end result of that procedure. Training is a procedure external to the network; learning is an internal process or activity. Training is done by example. In neural network research, there are three distinct ways of training.

The first and most common is **supervised training**. In this technique, the network is presented with a training set of input and output pairs: input pattern and corresponding desired output pattern. The learning law for such networks typically computes an error between the desired output and actual output of the network. This error is then fed back to modify the weights on the interconnections between the neuodes or layers.

Second: **reinforcement training** (also called graded training). This training is similar to supervised training except that an exact desired output is not provided. So reinforcement training is practised not by error but by "grade" on how well the network is doing. There are a number of schemes for this kind of training, varying from giving merely a brief "you

succeeded" or "you failed" message to more informative " too high" or " too low" performance feedback.

Thirdly and finally: **unsupervised training** or **self-organization**. In this procedure the network is presented only with a series of input patterns and is provided no information or feedback at all about its performance levels. Networks that use this kind of training procedure are most often used only for categorisation or statistical modelling applications because the network's specific responses cannot be predetermined by the designer.

Chapter 4

The Methodology Of Network Ensembles

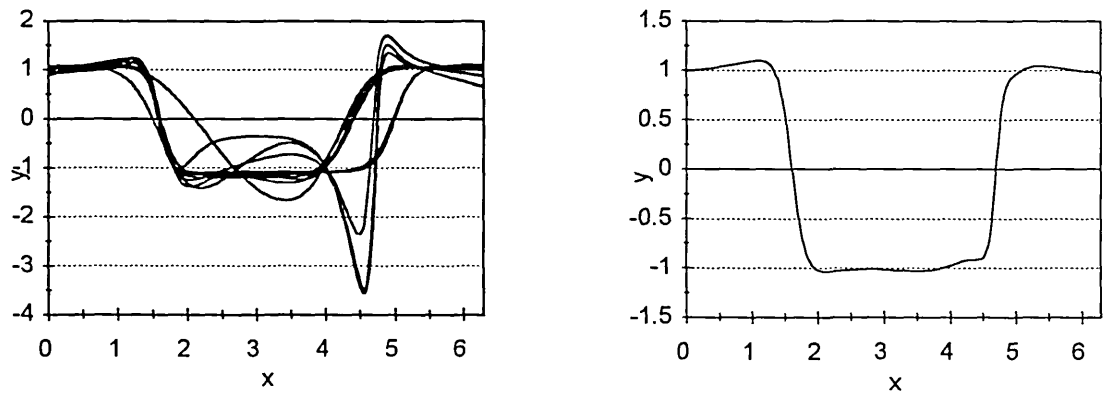
4.1 Introduction

In the earlier days of neural network research, researchers seemed to be satisfied with their achievements through the use of a single network [93]. More recently, it has been apparent that many tasks can be more effectively solved by employing a set of neural networks. There are a number of practical advantages to either decomposing a task into subtasks or combining several different solutions to the same task. In our study we will focus our interest on the latter. Through combining several different neural networks (a hybrid system) trained on the same task, the reliability and accuracy of the neural network generalisation can be improved.

In the neural network community, such a combined set of different networks is referred to as an ensemble. By using an ensemble, outputs of the ensemble networks are combined in some way. The aim is to obtain a more reliable and accurate ensemble output than either would be obtained by training a single network or by selecting the best network out of several. The use of an ensemble can provide an effective alternative to the tradition of generating a population of networks, and then choosing the one with the best performance, while discarding the rest. The basic idea underlying the ensemble-based approach is to find ways of fully exploiting, instead of ignoring, the information contained

in these redundant networks. Figure 4.1-1 indicates that although the outputs of the 10 networks disagree considerably, the output obtained by averaging them is quite desirable.

Combining estimators to improve performance has quite a long history, although it has recently received more attention in the neural network community. Research on combining estimators in neural network computing can be traced back to Nilsson [78], and is found



(a) The outputs of 10 networks being trained to approach the square wave

(b) The combined output from averaging the outputs of the 10 networks

Figure 4.1-1: Although the outputs of the 10 networks considerably disagree, the combined output from averaging them is quite desirable.

in a number of fields such as econometrics [19][41]; machine learning [5] and software engineering [58][66]. In the neural network community, ensembles of neural networks have been investigated by several authors [43][108][84] [59]. Recently, it has emerged that generalisation performance can often be improved by training not just one predictor, but rather using an ensemble, *i.e.*, a finite collection of a number of predictors, all trained for the same task. This idea of improving generalisation performance by combining the predictions of many different predictors has been investigated extensively in statistics [41] [108] [13][15][16]. Within the context of neural network learning, ensembles have also been studied by several groups [43][84][44][59][60]. Usually the predictors in the ensemble are trained independently and then their predictions are combined. This combination can be done by majority (in classification) or by simple averaging (in

regression), but one can also use a weighted combination of the predictors. We will focus on the simple averaging method in our future experiments so as to simplify the models' combination and reduce computing costs. Other schemes for combining predictors exists, like mixtures of experts [53] where the weighting of the ensemble members is highly non-linear, and boosting [37] in which the training data are partitioned among the individual predictors in a way that optimises the ensemble performance.

Ensemble methods, combined with active learning, which we will present in and after Chapter 6, provide an effective means for neural network training. An important point is that it is possible to formulate the problem of ensemble methods at an abstract level, as it is applicable to a much wider variety of statistical methods, educational and sociological behaviours.

4.2 The Ensemble Method

4.2.1 The Basic Ensemble Method (BEM)

In this section we present a basic ensemble method. The basic ensemble method is to combine a population of regression estimates to estimate a function $g(x)$ defined by $g(x)$.

Suppose that we have two finite data sets whose elements are all independently and identically distributed random variables: a training data set $D = \{(x_1, y_1), (x_2, y_2), ..., (x_p, y_p)\}$ and a cross-validatory data set $CV = \{(x_1^*, y_1^*), (x_2^*, y_2^*), \ldots, (x_q^*, y_q^*)\}$. Further, suppose that we have used D to regulate and generate a set of functions, $F = f_i(x)$ $(i = 1, 2, \ldots N)$, each element of which approximates $g(x)$. We would like to find the best approximation to $f_i(x)$ using D.

Define the misfit of function $f_i(x)$, the deviation from the target solution, as $m_i(x) = g(x) - f_i(x)$. The mean squared error (MSE) can now be written in terms of $m_i(x)$ as

$$\mathrm{MSE}[f_i] = \mathrm{E}[\mathrm{m}_i^2]. \tag{4.1}$$

The average mean squared error is therefore

$$\overline{MSE} = \frac{1}{N}\sum_{i=1}^{i=N} \mathrm{E}[\mathrm{m}_i^2].$$

Define the BEM regression function $f_{BEM}(x)$, as

$$f_{BEM}(x) = \frac{1}{N}\sum_{i=1}^{i=N} f_i(x) = g(x) - \frac{1}{N}\sum_{i=1}^{i=N} \mathrm{m}_i(x).$$

If we now assume that the $\mathrm{m}_i(x)$ are mutually independent with zero mean, we can calculate the mean squared error of $f_{BEM}(x)$ as

$$\mathrm{MSE}[f_{BEM}] = \mathrm{E}\left[\left(\frac{1}{N}\sum_{i=1}^{i=N} \mathrm{m}_i\right)^2\right] = \frac{1}{N^2}\mathrm{E}\left[\sum_{i=1}^{i=N} \mathrm{m}_i^2\right] + \frac{1}{N^2}\mathrm{E}\left[\sum_{i\neq j} \mathrm{m}_i \mathrm{m}_j\right]$$

$$= \frac{1}{N^2}\mathrm{E}\left[\sum_{i=1}^{i=N} \mathrm{m}_i^2\right] + \frac{1}{N^2}\sum_{i\neq j} \mathrm{E}[\mathrm{m}_i]\mathrm{E}[\mathrm{m}_j]$$

$$= \frac{1}{N^2}\mathrm{E}\left[\sum_{i=1}^{i=N} \mathrm{m}_i^2\right],$$

which implies that

$$\mathrm{MSE}[f_{BEM}] = \frac{1}{N}\overline{MSE}. \tag{4.2}$$

This is a powerful result because it theoretically justifies that by averaging regression estimates, we can reduce the mean squared error by a factor of N when compared to the population performance. By increasing the population size, we can in principal make the estimation error arbitrarily small! In practice, however, as N becomes large the assumption that the misfits, $\mathrm{m}_i(x)$ are mutually independent with zero mean, may not hold. In certain cases where we have nearly duplicate networks in an ensemble system, we are likely to have nearly linearly dependent misfits $\mathrm{m}_i(x)$. To deal with correlated or linearly dependent networks, many authors provide efficient solutions [84] [45][28][17][104]. In most cases, the number of member networks used in an ensemble is no more than 20 [28][82][84] [60] etc.

Let us consider the individual elements of population F. These estimators will more or less follow the true regression function. If we think of the misfit functions as random noise functions added to the true regression function and these noise functions are uncorrelated with zero mean, then the averaging of the individual estimates is like averaging over the noise. In this sense, the ensemble method is smoothing in the network function space F and can be thought of as a regularizer with a smoothness assumption on the true regression function [84].

An spin-off of ensemble methods is that regression estimates can come from various sources, which provides great flexibility in the application of ensemble methods. For instance, the neural networks can have different architectures, or be trained by different training algorithms or be trained on different training examples. We will demonstrate these in later chapters.

4.2.2 Bias/Variance Trade-off for An Ensemble of Predictors

To further understand the purpose of ensemble methods, we now are considering decomposition of mean squared error formula (4.1). The motivation to our approach follows from a key observation regarding the bias variance decomposition, namely the fact that ensemble averaging does not affect the bias portion of the error, but reduces the variance when the estimators on which averaging is done are independent.

The classification problem is to estimate a function $f_D(x)$ of observed data characteristics x, predicting a class labelled y, based on given a training set $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_p, y_p)\}$ using some measure of the estimation error on D. A good estimator will perform well not only on the training set, but also on new validation sets which were not used during training. Evaluation of the performance of the estimator is commonly done via the mean squared error (MSE) by taking the expectation with respect to the (unknown) probability distribution p of y:

$E[(y - f_D(x))^2 | x, D]$

This can be decomposed into

$$E[(y - f_D(x))^2 | x, D] = E[(y - E[y|x])^2 | x] + E[(E[y|x] - f_D(x))^2 | x, D]$$

The first term does not depend on the training data D or on the estimator $f_D(x)$; it measures the amount of "noise" or variability of y given x. Hence, f can be evaluated by using $E[(E[y|x] - f_D(x))^2 | x, D]$.

The empirical MSE of f is given by

$$E_D[(E[y|x] - f_D(x))^2]$$

where E_D represents expectation with respect to all possible training sets D of fixed size.

we can decompose the error to bias and variance components to get

$$E_D[(E[y|x] - f_D(x))^2] = (E_D[f_D(x)] - E[y|x])^2 + E_D[(f_D(x) - E_D[f_D(x)])^2] \quad (4.3)$$

The first term on the right-hand side of the equation(4.3) is called the bias of the estimator and the second term is called the variance. When training on a fixed training set D, reducing the bias with respect to this set may increase the variance of the estimator and contribute to poor generalisation performance. This is known as the trade-off between variance and bias. Typically, variance is reduced by smoothing; however, this may introduce bias (since, for example, it may blur sharp peaks). Bias is reduced by prior knowledge. When prior knowledge is used also for smoothing, it is likely to reduce the overall MSE of the estimator.

When training neural networks, the variance arises from two terms. The first term comes from inherent data randomness and the second term comes from the nonidentifiability of the model, namely, the fact that for given training data, there may be several (local) minima on the error surface.

Consider the ensemble average $f_{BEM}(x) = \frac{1}{N}\sum_{i=1}^{i=N} f_i(x)$

These predictors are identically distributed and, thus, the variance contribution (equation (4.3)) becomes (we omit x and D for clarity)

$$\mathrm{Var}(f_{BEM}) = \mathrm{E}[(f_{BEM} - \mathrm{E}[f_{BEM}])^2] = \mathrm{E}[\ (\frac{1}{N}\sum_{i=1}^{i=N} f_i - \mathrm{E}[\ \frac{1}{N}\sum_{i=1}^{i=N} f_i\ ]\ )^2\ ] \quad (4.4)$$

After some operations, equation (4.4) becomes (see [90]):

$$\mathrm{Var}(f_{BEM}) = \mathrm{E}[(f_{BEM} - \mathrm{E}[f_{BEM}])^2]$$

$$= \frac{1}{N^2}\sum_{i=1}^{i=N} \{\mathrm{E}[f_i^{\ 2}] - (\mathrm{E}[f_i])^2\} + \frac{2}{N^2}\sum_{i\neq j} \{\mathrm{E}[f_i f_j] - \mathrm{E}[f_i]\mathrm{E}[f_j]\} \quad (4.5)$$

From equation (4.5), if the predictors $\{f_i\}$ are highly correlated, for example if $f_i = f_j = f$ for all i, j, then the above equation becomes

$$\mathrm{Var}(f_{BEM}) = \frac{1}{N}\mathrm{Var}(f) + \frac{2}{N^2}\frac{N(N-1)}{2}\mathrm{Var}(f) = \mathrm{Var}(f)$$

Namely, there is no reduction in ensemble variance, $\mathrm{Var}(f_{BEM})$. On the other hand, If the predictors are identically distributed and independent, then the second term drops out and we are left with

$$\mathrm{Var}(f_{BEM}) = \frac{1}{N}\mathrm{Var}(f_i) \quad (4.6)$$

This result is similar to the one in the last section, *i.e.*, $\mathrm{MSE}[f_{BEM}] = \frac{1}{N}\overline{MSE}$, viewed from another standpoint.

The success of ensemble averaging of neural networks in the past[14][43][84][108] is due to the fact that neural networks have in general many local minima, and thus even with the same training set, different local minima are found when starting from different random initial conditions (for example, random initial weights). These different local minima lead to somewhat independent predictors, and thus the averaging can reduce the variance. When a larger set of independent networks is used, but no more data are available, data reuse methods can be useful. Bootstrapping [16] has been very helpful, since by resampling from the training examples, the independence of the training sets is increased, and hence, the independence of the estimators, leading to improved ensemble results. We will demonstrate this in later chapters.

4.3 Methods for Creating Ensemble Members

As mentioned above, as N (the number of ensemble networks) becomes large the assumption that misfits, $\mathrm{m}_i(x)$ mutually independent with zero mean, may not hold. In addition, ten identical copies of the same stock exchange forecast obviously contain exactly the same amount of information as just one copy. However, by obtaining ten different forecasts, it may actually be possible to predict tomorrow's stock information more accurately, even if the forecasts are all based on the same stock data. The same is true quite generally for ensemble learning; only if the predictors in an ensemble are different is there something more to be gained from using an ensemble.

There are a number of methods by which one can generate an ensemble system consisting of different networks. These include the topology of the networks (e.g. varying hidden units), initial random weights, the training examples, and training algorithms. We will provide an overview of the main methods used for creation of ensemble members. In

practice we usually use several methods simultaneously in order to reach the goal of killing two birds with one stone: both to avoid misfits $m_i(x)$ correlated in functional space and to force ensemble members to be as different as possible.

1. Varying initial random weights: a set of networks can be created by varying the initial random weights while holding the training example constant.

2. Varying the topology of neural networks: a set of networks can be created by varying the topology or architecture, or training with a varying number of hidden units while holding the training example constant.

3. Varying the training algorithm used: The algorithm used to train the networks could be varied while holding the training example constant.

4. Varying the training examples: This method seems to be most frequently used, involving altering the training examples.

In the work described in this thesis, we employ a combination of three of the above techniques (1, 2, 4) in our experiments.

4.4 Sampling Data

Varying training examples is referred to as sampling data. A common approach to the creation of a set of networks for an ensemble is to use some form of sampling technique, such that each network in the ensemble is trained on a different subsample of the training examples. Resampling methods which have been used for this purpose include cross-validation [59][98] and bootstrapping [16][89][90][30].

4.4.1 A Bootstrap Ensemble

The bootstrapping technique has become one of the major tools for producing empirical confidence intervals of estimated parameters or predictors[31][16][80][89][90]. A simple bootstrap procedure amounts to sampling with replacement from the training data, and constructing several training subsets, all with the same size but data randomly selected from the original training set[89]. Later, the variability between the estimated parameters can be measured, and give some indication about the true variability of the model parameters arising from the data, Furthermore, variability of the prediction, or error bars on the prediction, can also be estimated in this way.

Consider a total of P items available in the original training set S. The approach is to generate N training subsets $\{S^1, \ldots, S^N\}$ from the original training set, each containing some percentage of the original training set S. The subsets are constructed independently from the original training set by drawing at random with replacement [31]. Each subset is used to train a network in the ensemble.

Using this bootstrap procedure, each subset is expected to include roughly 36% ~ 60% duplicates (due to replacement during sampling) [82]. There are two important issues in training neural networks to develop a bootstrap ensemble. The first is whether to use all P items (including duplicates) in the training subsets S^i or to use only unique (non-duplicate) items. If only a distinct fraction is used for training, the size of the training subset would be less than P. The second issue concerns the leftover fraction: it can be used as an early-stopping validation set or just to let the network over-fit to the training subsets. Early stopping usually requires a fraction of the data to be taken from the original training set, which might degrade the performance of the neural network. The advantage of a bootstrap ensemble is that the leftover sample is already available.

Our early experiments suggest that using only a distinct fraction of the data and letting the network over-fit to the training subsets increases the independence of the predictors in the ensemble, and thus improves the generalisation performance.

The Bootstrap Algorithm

1) Generate bootstrap subsets $\{S^1, \ldots, S^N\}$ from the original training set, where N is the number of networks in the ensemble.

2) For each bootstrap subset, collect unsampled items into leftover sample sets, giving $\{s^1, \ldots, s^N\}$.

3) For each S^i, train a network. Use the leftover set s^i as a validation stopping criterion if necessary. This gives N neural network predictors: $\{f(x; S^1), \ldots, f(x; S^N)\}$.

4) Build a ensemble from the N bootstrap networks using a simple averaging procedure:

$$f_{BEM}(x) = \frac{1}{N} \sum_{i=1}^{i=N} f(x; S^i).$$

Alternatively, If weighted averaging is used, estimate the weights of the networks α_i using the entire training set and calculate the output of the ensemble as

$$f_{BEM}(x) = \sum_{i=1}^{i=N} \alpha_i f(x; S^i)$$

4.4.2 A Cross-Validation Ensemble

The algorithm is quite similar to the procedure used in prediction error estimation. First, generate subsets from the original training set by sampling without replacement removing a fraction of the original training set. Let S denote the original training set, and S^i denote a subset (with s-fraction removed from S, $S^i = S - s^i$). There are two versions of cross-validation commonly used: delete-one and hold-out. In the case of delete-one cross-

validation, N subsets are constructed. Only the hold-out method will be considered in our study, for obvious reasons (i.e. costly computations for training N networks).

The procedure starts with deciding the number N of networks in the ensemble. Divide the original training set S into N fractions of roughly equal size. An important issue in the cross-validation ensemble is the degree of data overlap between the subsets $\{S^1, \ldots, S^N\}$. We have to decide how large each subset S^i is compared with the original training set S. The degree of overlap depends on the number of subsets (the size of ensemble, N) and the size of a removed fraction s^i (and the size of S^i) from the original sample. Normally, the size of removed fraction s^i (the validation sets) is equal to S/N. In this case, each validation set contains no same data without overlap.

Like the bootstrap ensemble, there is a fraction from the original training set that is not included in each training subset S^i. This unused fraction can be used as a validation criterion for early stopping. Cross-validation ensembles fit nicely into the practice of neural network modelling where removing a fraction of the data to stop training is common.

The Cross-Validation Ensemble Algorithm

1) Divide the original training set S into N fractions $\{L_1, \ldots, L_N\}$. The choice of N depends on the number of networks in the ensemble and the degree of overlap desired.

2) Decide how many L will be used to construct each training subset S^i. Suppose r (r is the number of cross-validation sets in L_j, $r < N$), $S^i = \sum_{j=1}^{r} L_j$ and $s^i = S - S^i$.

3) Construct each training subset S^i by picking r fraction of L from the original training set S, using a "revolving door procedure" [82].

4) Train a network $f(x; S^i)$ using each training subset S^i.

5) If an early-stopping procedure is used, use the leftover fraction s^i as an early-stopping criterion.

6) Build an ensemble from the constructed networks using the simple averaging procedure:

$$f_{BEM}(x) = \frac{1}{N}\sum_{i=1}^{i=N} f(x; S^i).$$

Alternatively, if weighted averaging is used, estimate the weights of the networks α_i using the entire training set and calculate the output of the ensemble as

$$f_{BEM}(x) = \sum_{i=1}^{i=N} \alpha_i f(x; S^i)$$

The fraction of data overlap determines the trade-off between individual network performance and error correlation between the networks. Lower correlation can be expected if the networks train with less overlapped data, which means a larger removed fraction and smaller fraction for training. The smaller the training subset size, the lower the individual network performance that can be expected.

4.5 Methods of Combining Ensemble Networks

Once a set of networks has been created, an effective way of combining their several outputs must be found. There are several different methods of combining, and since a number of reviews of the topic already exist [40][54][109][113], we will do no more than briefly outline some of the more common methods.

4.5.1 Averaging and Weighted Averaging

Linear opinion pools are one of the most popular aggregation methods, and refer to the linear combination of the outputs of the ensemble members' distributions with the

constraint that the resulting combination is itself a distribution [54]. A single output can be created from a set of network outputs via simple averaging [84] or by taking a weighted average [44][84].

4.5.2 Non-linear Combining Methods

Other non-linear combining methods that have been proposed include Dempster-Shafer belief-based methods [92], combining using rank-based information [2], voting [43] and order statistics [103].

4.5.3 Supra Bayesian

Jacobs [54] contrasts supra Bayesian with linear combinations. The underlying philosophy of the supra Bayesian approach is that the opinions of the experts are themselves data. Therefore, the probability distribution of the experts can be combined with its own prior distribution.

4.5.4 Stacked Generalisations

Under stacked generalisation [108], a non-linear network learns how to combine the networks with weights that vary over the feature space. The outputs from a set of level 0 generalizer are used as the input to a level 1 generalizer, which is trained to produce the appropriate output. The term "stacked generalisation" is used by Wolpert [108] to refer both to this method of stacking classifiers and also to the method of creating a set of ensemble members by training on different partitions of the data. It is also possible to view other methods of combining, such as averaging, as instances of stacking with a simple level 1 generalizer. The same idea has been adapted to regression tasks, where it is termed "stacked regression" [13]. A comprehensive exploration of stacking is reported by LeBlanc and Tibshirani [62].

Using an ensemble of classifiers, instead of a single classifier, can lead to improved generalisation. The combination of the ensemble methods and active learning techniques

provides a very useful and powerful tool for our practical problems. In the following chapters, we will demonstrate these methods in detail.

Chapter 5

Neural Network Ensembles, Cross-Validation and Active Learning Using Discrepancy

5.1 Overview

In the last chapter we have presented a theoretical discussion that neural network ensembles (committees) can give improved accuracy and reliable estimation of predictors. In this Chapter we will now demonstrate a practical application with an experiment from the standpoint of how to improve generalisation. We define the "discrepancy" as the variance of the output of ensemble members averaged over unlabeled data, which quantifies the disagreement among the networks. We discuss how to use this discrepancy in combination with cross-validation to give a reliable estimate of the ensemble generalisation error, and how this type of cross-validation ensemble can sometimes improve generalisation performance. Finally, by a generalisation of "query by committee", we show how the discrepancy can be used to select new training data to be labelled in an active learning scheme.

5.2 Neural Network Ensembles

We have discussed in Chapter 4 how a combination of many different networks can improve generalisation performance. In practice, the networks in the ensemble are usually trained individually and then their outputs are combined. This combination is usually done by majority (in classification) or by simple averaging (in regression).

Our study indicates that a combination of the output of several networks (or other predictors) is only useful if they disagree on some inputs. Clearly, there is no more information to be gained from a million identical networks than there is from just one of them [102][16][45][54][109][60][59]. By quantifying the disagreement in the ensemble it turns out to be possible to state this insight rigorously for an ensemble used for the approximation of a real valued function (In this experiment we use a square wave function). The method of cross-validation was therefore introduced so as to train the networks on different training sets.

We define discrepancy as the variance of the output of ensemble members averaged over the unlabeled data, so it quantifies the disagreement among the networks. The discrepancy in combination with cross-validation is used to give a reliable estimate of the ensemble generalisation error. This type of cross-validation ensemble can sometimes improve performance.

Assume the task is to learn a function g from R^N to R for which one has a sample of p examples, $D = \{(x_1, y_1), (x_2, y_2), ..., (x_p, y_p)\}$ where $y_u = g(x_u)$ and $u = 1, 2, ..., p$. These examples are assumed to be drawn randomly from the input space $p(x)$. Anything in the following is easy to generalise to several output variables.

Assume the ensemble consists of N networks and the output of network i on input x is called $f_i(x)$. A weighted ensemble average is denoted by a bar, like

$$\bar{f}(x) = \sum_{i=1}^{i=N} \alpha_i \, f_i(x) \tag{5.1}$$

This is the final output of the ensemble, namely, $f_{BEM}(x)$. We think of the weight α_i as our belief in network i and therefore constrain the weights to be positive and sum to one. The constraint on the sum is crucial for some of the following results.

The *discrepancy* on input x of a single member of the ensemble is defined as

$$d_i(x) = (f_i(x) - \bar{f}(x))^2 \tag{5.2}$$

The *ensemble discrepancy* on input x is

$$\bar{d}(x) = \sum_{i=1}^{i=N} \alpha_i \, d_i(x) = \sum_{i=1}^{i=N} \alpha_i (f_i(x) - \bar{f}(x))^2 \tag{5.3}$$

This is simply the variance of the weighted ensemble around the weighted mean, and it measures the disagreement among the networks on input x. The quadratic error of network i and of the ensemble are

$$e_i(x) = (g(x) - f_i(x))^2 \tag{5.4}$$

$$e(x) = (g(x) - \bar{f}(x))^2 \tag{5.5}$$

where $g(x)$ is the target output of the function.

Respectively, adding and subtracting $g(x)$ in (5.3) yields

$$\bar{d}(x) = \sum_{i=1}^{i=N} \alpha_i \, e_i(x) - e(x) \tag{5.6}$$

We refer the weighted average of the individual error as $\bar{e}(x) = \sum_{i=1}^{i=N} \alpha_i \, e_i(x)$, thus this becomes

$$e(x) = \bar{e}(x) - \bar{d}(x) \tag{5.7}$$

Further, all these formulas can be averaged over the input distribution. Averages over the input distribution will be denoted by capital letters, obtaining following equations [59]

$$E_i = \int \mathrm{d}x p(x) \, e_i(x) \tag{5.8}$$

$$A_i = \int \mathrm{d}x p(x) \, d_i(x) \tag{5.9}$$

$$E = \int \mathrm{d}x p(x) \, e(x) \tag{5.10}$$

The first two of these are the generalisation error and the discrepancy respectively for network i, and E is the generalisation error for the ensemble. From (5.7) we then find for the *ensemble generalisation error*

$$E = \bar{E} - \bar{A} \tag{5.11}$$

The first term on the right is the weighted average of the generalisation errors of the individual networks ($\bar{E} = \sum_{i=1}^{i=N} \alpha_i \, E_i$), and the second is the weighted average of the discrepancies ($\bar{A} = \sum_{i=1}^{i=N} \alpha_i \, A_i$), which we refer to as the ensemble discrepancy.

The beauty of this equation is that it separates the generalisation error into a term that depends on the generalisation errors of the individual networks and another term that contains *all correlations* between the networks. Furthermore, the correlation term $\bar{A}$ can be estimated entirely from *unlabeled data*, *i.e.*, no knowledge is required of the real

function to be approximated. The term "unlabeled example" is borrowed from classification problems, and in this context it means an input x for which the value of the target function $f(x)$ is unknown.

Equation (5.11) expresses the trade-off between bias and variance in the ensemble again, but we present it in another different way than the common bias-variance relation [39] in which the averages are over possible training sets instead of ensemble averages. If the ensemble is strongly biased the discrepancy will be small, because the networks implement very similar functions and thus agree on inputs even outside the training set. Therefore the generalisation error will be essentially equal to the weighted average of the generalisation errors of the individual networks. If, on the other hand, there is a large variance, the discrepancy is high and in this case the generalisation error will be smaller than the average generalisation error [75].

From this equation we can immediately see that the generalisation error of the ensemble is always smaller than the (weighted) average of the ensemble error, $E < \overline{E}$. In particular for uniform weights:

$$E \leq \frac{1}{N} \sum_{i=1}^{i=N} E_i \tag{5.12}$$

which is analogous to the previous results as discussed in Chapter 4.

5.3 The Cross-Validation Ensemble

From (5.11) it is clear that increasing the discrepancy (while not increasing individual generalisation errors) will improve the overall generalisation. The more the networks disagree, the better the generalisation performance. Therefore we want the networks to disagree as much as possible! One way is to use different types of approximations like a mixture of neural networks of different topologies or a mixture of completely different

types of approximators. Another obvious way is to train the networks on different training sets. Here we use *cross-validation*. Or we can simultaneously employ two ways.

Choose a number $K \leq p$ [11]. For each network in the ensemble hold out K examples for testing, where the N test sets should have minimal overlap, i.e., the N training sets should be as different as possible. If, for instance, $K \leq p/N$ it is possible to choose the K test sets with no overlap. This enables us to estimate the generalisation error E_i of the individual members of the ensemble, and at the same time make sure that the discrepancy increases. When holding out examples the generalisation errors for the individual members of the ensemble, E_i, will increase, but the conjecture is that for a good choice of the size of the ensemble (N) and the test set size (K), the discrepancy will increase more and thus we will get a decrease in overall generalisation error.

5.4 Experiments With Neural Networks

In order to study the feasibility of the cross-validation ensemble method, we experimented with the square wave function (Figure 5.4-1). The ensemble consists of five feedforward networks, which are trained independently by backpropagation using 200 examples randomly produced from the input space of [-4, +4]. The architecture of the networks is composed of one hidden layer with 20 hidden units, all initialised with random weights.

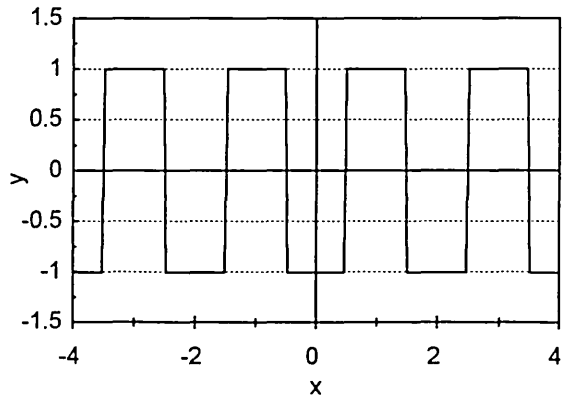


Figure 5.4-1: Target $f(x)$: The Square Wave

5.4.1 Cross Validation Technique

Now the ensemble of 5 networks is employed using 200 random examples for training, according to the Section 5.3, i.e., $K \leq p/N$, we choose K = 40 (200/5) so that each of the five test sets can be 40 examples with no overlap (see Table 5.4.1-1). Consequently, each network has a cross-validation set of size 40 and each is trained on the remaining 160 examples.

Table 5.4.1-1: Cross-validation sets of size 40 without overlap

net1	CV1	Training Set			
net2	Training Set	CV2	Training Set		
net3	Training Set		CV3	Training Set	
net4	Training Set			CV4	Training Set
net5	Training Set				CV5
1	40	80	120	160	200

Cross-validation is a suitable strategy for improving generalisation in networks of non-optimal size whose aim is to avoid "overtraining" by carefully monitoring the evolution of the validation error during training and stopping just before it starts to increase. This strategy is based on one of the early stopping criteria in model evaluation [98].

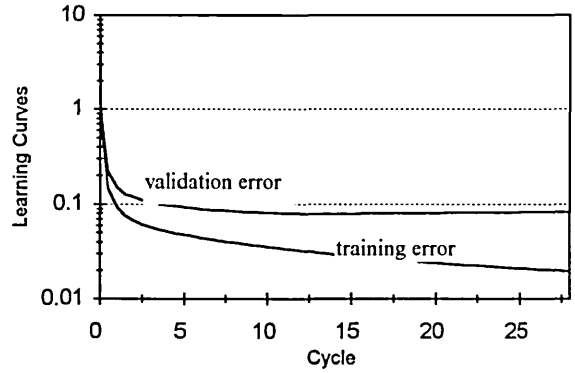


Figure 5.4.1-1: Training error and cross-validation errors of network 1 using backpropagation algorithm. The cycle unit 1×10^{3}.

Here, the whole available data set is split into two parts: training set and validation set. The training set is used to determine the values of the weights of each network in the ensemble. The validation set is used to for deciding when to terminate training. Training continues as long as the performance on the validation set keeps improving. When it ceases to improve, training is stopped. Figure 5.4.1-1 illustrates the training error and validation error of network 1, whose training cycles (stopping) is 191000. Of course, the other 4 networks have different training cycles (stopping). The outputs of the individual networks are shown in Figure 5.4.1-2, and the final ensemble output (by averaging the outputs of five individual networks), variance and root squared error are shown in Figure 5.4.1-3, from which it is shown that cross-validation technique can get the individual network outputs to disagree considerably and the combined output of the ensemble is desirable. Also it can been seen that the discrepancy curve has higher peaks near the discontinuities, which will be used for an active learning scheme for judiciously selecting training data in the next section.

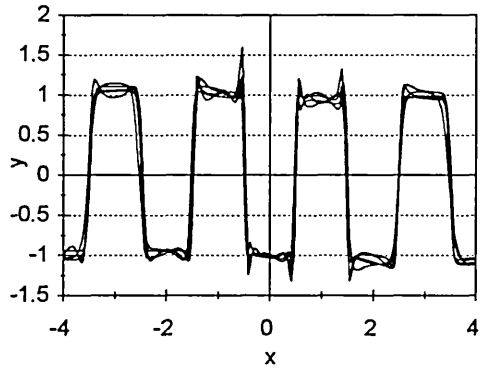


Figure 5.4.1-2: The outputs of the five individual networks. An ensemble of five networks were trained to approximate the square wave target function $f(x)$

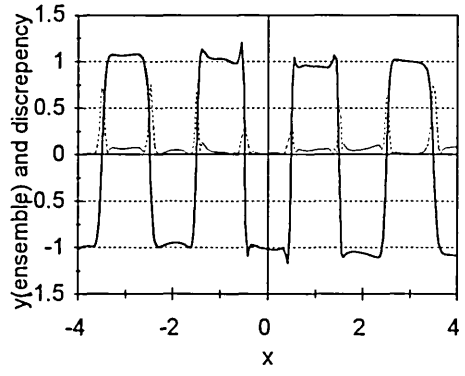


Figure 5.4.1-3: The final ensemble output (the solid line), and discrepency (or variance, the broken line).

5.4.2 Active Learning Using Maximum Discrepancy

In last section, we discussed how the ensemble method can give improved generalisation. Now we present a query-based active learning scheme that applies to function approximation. It is essentially a generalisation of query by committee [35] [95] that was

developed for classification problems. Our method is based on the conjecture that those patterns in the input space yielding the largest error are those points we would benefit the most from including in the training set.

Since the generalisation error is always non-negative, we see from (5.7) that the weighted average of the individual network errors is always larger than or equal to the ensemble discrepancy,

$$\bar{e}(x) \geq \bar{d}(x) \tag{5.13}$$

which tells us that the discrepancy is a lower bound for the weighted average of the squared error. An input pattern that yields a large discrepancy will always have a large average error. On the other hand, a low discrepancy does not necessarily imply a lower error. If the individual networks are trained to a low training error on the same set of examples then both the error and the discrepancy are low on the training points. This ensures that a pattern yielding a large discrepancy cannot be in the close neighbourhood of a training example. The discrepancy will to some extent follow the fluctuations in the error.

An ensemble consisting of 10 networks is trained to approximate the square-wave function shown in Figure 5.4-1, but in this experiment the function was restricted to the interval from -2 to +2. In addition, the feed-forward networks have different hidden units generated randomly, varying from 10 to 30, so that members of the ensemble have different architectures. We start with the same initial training set of 1 example chosen randomly from the input space of [-2, +2]. Examples are generated and added one at a time in following two learning schemes. The first is to randomly select exemplars for the training set; the second is to select each exemplar that gives the largest discrepancy (variance) out of 200 random data points in the input space [-2, +2]. The outputs of 10 network models and the final combined output by averaging 10 networks of the ensemble is seen in Figure 5.4.2-1. The generalisation performance (root mean squared error) via the training set size is seen in Figure 5.4.2-2. From Figure 5.4.2-1, it is clearly shown that although the individual network outputs are not ideal, the final combined output of the ensemble is desirable. Figure 5.4.2-2 tells us that the generalisation can be significantly improved by active learning, since active learning judiciously selects training

data while passive learning randomly selects training data. Thus, not only the ensemble method can improve generalisation, but also active learning can obtain better generalisation. In the next chapter we will discuss how active learning can dramatically reduce the size of the training data set as well.

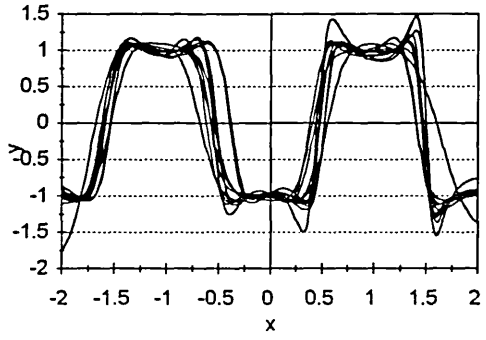


Figure 5.4.2 -1(a) : The outputs of 10 network models

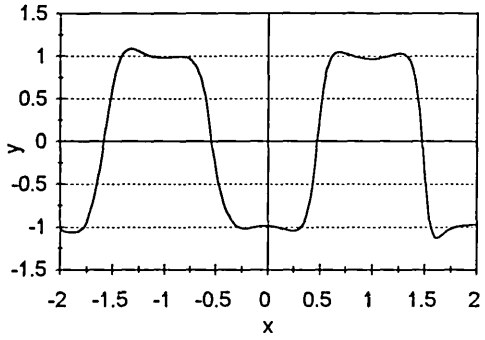


Figure 5.4.2-1(b): Final output by averaging 10 outputs of the ensemble

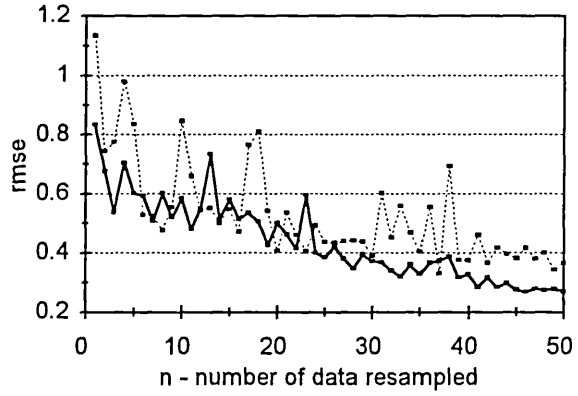


Figure 5.4.2 - 2: The solid line indicates the active learning generalisation curve, and the broken line indicates the passive learning generalisation curve based on the same training set size.

5.5 Conclusion

The central idea in this chapter is to show that there is a lot to be gained from using unlabelled data when training in ensembles. Although we deal with neural networks, this idea can be extensively applied for any other type of method used as the individual members of the group or ensemble.

It is shown that apart from getting the individual networks of the ensemble to generalise well, it is important for the individual members to disagree as much as possible in order to obtain better overall generalisation. This is done by training the individual networks on different training sets, and creating different individual networks (through producing random weights and different architectures).

Finally, a method for active learning is described, which is based on the method of query by committee developed for classification problems. The idea is that if the ensemble disagrees strongly on a data point, this data point will be the most needed information and should be included in the training set for ensemble learning. It is shown that active learning can improve generalisation performance.

In this chapter, we have discussed ensemble and active learning methods from the standpoint of generalisation. It will be discussed more generally in the next chapters that ensemble and active learning methods can not only improve generalisation, but also can improve data selection and learning efficiency as well. The active learning method discussed in this chapter is referred to as Maximum Variance because training data resampled depends on maximum variance of the ensemble state. In Chapter 6, we will develop a new method for active learning, by which training data selection depends on maximum inter-quartile range (MIQR) of the ensemble state. Because the MIQR method's philosophical idea is to embody the "majority's" desire by pruning the "minority" (outliers), training data resampled will be more informative than using the Maximum Variance method, therefore it is more effective and compares favourably with the Maximum Variance method (details will be demonstrated in the following chapters).

Chapter 6

MIQR Active Data Selection with Network Ensembles

6.1 Overview

In the last chapter we have demonstrated how to use "discrepancy" to select some particular training exemplars so as to improve generalisation performance. In this chapter we formally derive a method for selecting exemplars for training a set of networks with one-hidden layer feedforward architecture to approach functions. This method incrementally adds to the training set as necessary to achieve the desired level of accuracy. Our selection criterion does not depend on using a neural network estimator, thus it may be used for general purpose nonlinear regression using any statistical estimator.

The objective is to minimise the data requirement of learning and maximise improved generalisation as well. In a particular sense, we are performing a kind of data compression, by selecting exemplars representative of the set of all available examples. Towards this end, we choose a criterion for selecting training examples that works well in conjunction with creation of ensemble networks. We proceed sequentially, selecting an example that, when added to the previous set of training examples and learned, maximises the decrement of ensemble network squared error over the input space.

A number of criteria for selecting training examples have been proposed by many researchers. Among them, are Maximum Variance which we have discussed in the last

chapter (and other groups[59][89]), and Maximum Error[99][86][52][88] etc. Plutowski and White examine the learning task from a more general function approximation standpoint [86], viz., approximating a target function, $g(x)$, using a network output function, $f(x, w)$, parameterised by weights w. They design the criteria for selecting new examples to meet two objectives: (1) to optimise the accuracy of fit between network output, $f(x, w)$, and the target function, $g(x)$, and (2) to minimise the approximation's unreliability in the presence of noise. Their study quantifies the above two considerations by proposing an Integrated Mean Squared Error (IMSE) measure to be minimised. To select the next training example, the learning algorithm samples at the next input location x_{n+1} that maximally decreases the IMSE. Plutowski successfully implemented active learning in a single network system by using the Δ ISB criterion. However, his method is not only computationally complicated and expensive, but also strictly applicable only to a class of "reasonably smooth functions". Furthermore, Δ ISB is actually a variation of Maximum Error since Δ ISB derives from Integrated Mean Squared Error (IMSE) based on a Bayesian inference framework or Optimal Experimental Design. His method is not used in ensemble network systems, but mostly in a single network, where *labelled data* sets are provided. Another variation of Maximum Error is EISD(Expected Integrated Squared Difference) [99]. By minimising EISD, the next example can be located. The EISD can be successfully applied for "unknown" function approximation. It also is based on a Bayesian framework, being applicable to a single neural network. No matter whether the Δ ISB criterion or the EISD criterion, their theory is built up on the assumption that the function to be approximated is reasonably smooth, thus their methods are considerably restricted in practice.

We will propose the Maximum Inter-Quartile Range (MIQR) criterion for selecting training examples. MIQR is defined by computing the maximum inter-quartile range of the upper quartile and lower quartile of the outputs of ensemble networks. This idea is inspired by statistical methods. Using the MIQR criterion, example selection is not influenced by ***"outliers"***, rather, principally dependent upon the ***"mainstream"*** of the ensemble networks. Encompassed in our new method is a very simple ancient philosophical idea, *i.e.* ***"minority obey majority"***. Outliers are only a minority; on the other hand, the majority within the inter-quartile range stands for mainstream and the

general trend of the ensemble networks. Our experiments show that the MIQR criterion is an effective method for example selection and outweighs a contending method, the Maximum Variance criterion. The motivation of the use of MIQR instead of Maximum Variance is that maximum variance often results in local minima because variance computation is essentially based on the assumption that values are normally distributed. Neural networks, however, are essentially nonparametric, therefore values of the variance of network outputs are random.

We conclude with graphical illustrations of the method, and demonstrate its use during network training. Several benefits are apparent for practical use in a variety of applications. Experimental results indicate that training upon exemplars selected in this fashion can save computation in general purpose use as well.

We demonstrate that our method selects concise training sets, and that it can be used to improve generalisation performance. The method works well on networks with multiple inputs and outputs. Theoretically, it should work on problems of any dimension. However, experience in high dimensional domains is required to determine whether the benefits of our technique outweigh the overhead of the selection process. In addition, due to the MIQR method being computed on a basis of unlabelled data sets, our method can be easily and directly extended to be applicable to noisy data collection without difficulty.

6.2 Derivation of the Method

6.2.1 Outliers, Quartiles and Inter-Quartile Range

We recall the topic of "outliers" and "quartiles" in our preliminary work on developing a method for selecting exemplars. The study of "outliers" and "quartiles" is outside the scope of this dissertation. We consider these two terms to have many different connotations and interpretations. The reader interested in this topic may begin with the study of Robust Statistical Procedures given in [50] and Outliers in Statistical Data given in [6], as well the references provided in [55][11][67]. As a small peek inside this

actively studied topic, we provide two distinct views on the matter. The following quote depicts outliers as the result of finiteness of the sample:

> The term outliers does not mean that they are not part of the joint data probability distribution or they contain no information for estimating the regression surface; it means rather that outliers are too small a fraction of the observations to be allowed to dominate the small-sample behaviour of the statistics to be calculated. With parametric regression modelling techniques it is easy to quantify this effect by simply computing the effect that each data point has on the regression surface. This is not a trivial problem in non-parametric modelling but the statistics literature is full of methods to deal with it [83].

The following perspective views outliers as a result of the data generating process:

> Outliers are generated from a distribution that is a perturbation to the underlying distribution, for example, a small amount of noise with ever changing distribution in the background[67].

Yet another view is provided in [55], which uses a maximum-likelihood definition of "outliers." In our work, although we refer to pruned outputs of the members of an ensemble as "outliers," we simply excise outputs that are misleading, according to a estimate of how much an estimate of generalisation is improved by deleting them.

The terms "quartiles" and "inter-quartile range" can be also found in the references[24][6][27]. Here, we explain quartiles and range by using a rule. Suppose we have a set of ensemble networks (N), which have outputs $Y_1(x), Y_2(x), Y_3(x)$, ..., $Y_{N-1}(x)$, $Y_N(x)$ respectively, (see Figure 6.2.1-1), here, x is taken from the input space according to a certain distribution probability p. For any given x_i (i = 1, 2, 3, ..., p), we rearrange the outputs $Y_1, Y_2, Y_3, ..., Y_{N-1}, Y_N$ to ascending order of $Y_1^*, Y_2^*, Y_3^*, ..., Y_{N-1}^*, Y_N^*$ such that

$$Y_1^* \le Y_2^* \le Y_3^* \le \ , \ldots , \le \ Y_{N-1}^* \le \ Y_N^*$$

If N is odd, the ***median*** is defined as the $Y^*_{(N/2)+1}$. If N is even, the ***median*** actually is the middle position of the order $Y^*_{(N+1)/2}$. Then the ***upper quartile*** is defined as midway between (***median*** + 1) and Y^*_N; and the ***lower quartile*** as midway between (***median*** - 1) and Y^*_1. Finally the ***inter-quartile range*** is defined as the difference between the upper quartile and lower quartile.

For instance, assume we have the 15 outputs below:

2, 4, 11, **21**, 34, 65, 76, **88**, 89, 100, 126, **156**, 187, 201, 273

↑ lower quartile ↑ median ↑ upper quartile

Obviously, the median is 88; the lower quartile and upper quartile are 21 and 156 respectively; the values 2, 4, 11, and 187, 201, 273 are outside the quartiles and may include outliers.

This can be further illustrated by Figure 6.2.1-1, which is the outputs of the ensemble consisting of 10 networks approaching sin(x) in the fourth training iteration. Clearly, the value of the inter-quartile range (x) changes at each point in the input space.

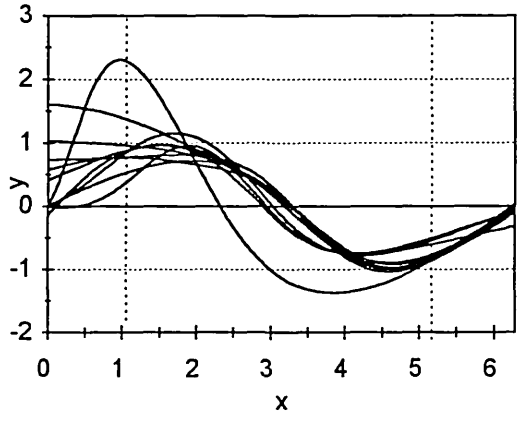


Figure 6.2.1-1: Outputs of the ensemble consisting of 10 networks. Quartiles change on each x point. For example, the quartiles near x = 1 are different from ones near x = 5. Correspondingly, the value of the inter-quartile range(x) changes on each x point.

6.2.2 MIQR for Selecting Examples

Now it is easy to present our method for selecting training exemplars. Suppose we have an ensemble which is composed of N feedforward networks with one-hidden layer. We create such an ensemble by simultaneously using ways introduced in Chapter 5: (1)varying initial random weights; (2) varying the topology of neural networks; and (3) varying the training examples, here using the bootstrapping technique.

Suppose we are provided with a set of P "candidate" examples generated randomly from the input space (here the X-axis): $\{(x_1, y_1), (x_2, y_2), ..., (x_p, y_p)\}$, and $y_i = g(x_i)$, $i = 1, 2, 3, ..., P$. We denote this set as x^P, Suppose we have a small subset consisting of n examples, x^n, randomly chosen from x^P (most often, we begin by putting $n = 10$) [89][81][16]. By using bootstrapping, a resampling technique, such a training set can be resampled into N subsets for training N networks. After each training iteration, we compute the ensemble inter-quartile range(x) for each $x \in x^P$. Obviously, the points of larger inter-quartile ranges are the points where more information is required. Namely, we choose

$$x_{n+1} = \arg\max_{x} \text{Inter-Quartile Range}\,(x \mid x^P). \tag{6.1}$$

Thus we derive a criterion for evaluating individual examples that allows us to select exemplars maximising the inter-quartile range that would result from adding the new exemplar to the training set.

In adding to the training set, a number of practical matters must be addressed. Because we are trying to learn a deterministic mapping, it is appropriate to specify a tolerance $\varepsilon_P^{\max}$ representing the desired accuracy for the fit over all of the candidates as measured by

$$\varepsilon_P = \left[\frac{1}{P} \sum_{x \in x^P} (g(x) - f_{ens}(x))^2 \right]^{1/2}. \tag{6.2}$$

here, $g(x)$ is the target function, $f_{ens}(x)$ is the average output of the ensemble networks, namely,

$$f_{ens}(x) = \frac{1}{N} \sum_{x \in x^P} f_i(x, w_n). \tag{6.3}$$

where $f_i(x, w_n)$ is the i th member of the ensemble, and w_n is the vector of weights when f_i has been trained by training set x^n. We desire $\varepsilon_P(x^n) \leq \varepsilon_P^{\max}$. Clearly, before selecting a new exemplar, we should always check whether $\varepsilon_P(x^n) \leq \varepsilon_P^{\max}$, and quit if it does. We refer to a set of exemplars that give this condition as "sufficient".

If this condition does not hold, we must determine when the current exemplars have been fitted well enough to select a new one, as the underlying theory and practical experience suggest that, for given network complexity, the selection criterion will work best if the fit is optimal for the current set of exemplars. Optimality for the given network complexity is ensured either by finding that the training threshold ε_n is sufficiently small, where

$$\varepsilon_n = \left[\frac{1}{n} \sum_{x \in x^n} (g(x) - f_i(x, w_n))^2 \right]^{1/2}, \tag{6.4}$$

or by finding that repeated optimisation attempts do not give further improvement in ε_n. In implementing the requirement that ε_n be sufficiently small, we specify that $\varepsilon_n \leq \varepsilon_p^{\max}$ when selecting x_{n+1}. This requirement follows from the observation that if the given model cannot adequately fit the training data points, then it is unlikely it will adequately fit the entire set of candidates, because $x^n \subset x^P$.

Our experience indicates that fitting x^n to an even closer tolerance can result in a more concise set of exemplars, i.e., a sufficient set of smaller size. When the minimal number of exemplars is desired, ε_n should be stringently minimised.

If a somewhat larger than minimal set of exemplars is acceptable, then it may not be necessary to obtain the globally minimal value for ε_n before selecting a new exemplar. The task of seeking the most concise training set can require more total overhead (which includes training as well as selection cost) than for another sufficient but larger set [88]. Relaxing the tolerance on ε_n allows a new exemplar to be selected sooner. This reduces

the cost of training over the exemplars, and can thus reduce total overhead as well even though more exemplars may need to be selected to comprise a sufficient set. We regulate this by using a variable ε_n^{max} to determine when ε_n is sufficiently small to select a new exemplar.

Before we can select a new exemplar, we require that each member network of the ensemble fits the current training set "sufficiently well". Sometimes, active selection chooses a new exemplar "too close" to or the same as previously selected exemplars. This is easy to detect, and in this case we reject the new exemplar and continue with training. We use an "exemplar spacing" parameter d to detect when a new exemplar is too close to or the same as a previous selection. Two examples x_i and x_j are "close" in this sense if they are within Euclidean distance d, and if additionally $|g(x_i)-g(x_j)| \leq \varepsilon_P$. The additional condition allows the new exemplar to be accepted even when it is close to a previous selection in the input space, provided it is sufficiently far away in the output space.

A problem that can arise during this process is that an exemplar may be selected repeatedly amongst or close to the previous training exemplar, here $x^n = \{x_1, x_2, ..., x_n\}$ and $n \leq p$. This can indicate that optimisation in the prior exemplars was not complete (i.e., ε_n can be reduced further,) but also is observed to occur when network complexity is insufficient to achieve $\varepsilon_n \leq \varepsilon_P^{max}$. Such exemplars repeatedly selected are easy to detect, and in such cases we simply discard the "new" exemplar and continue training on current training set. To avoid the cost of repetitively selecting a new exemplar only to reject it, we modify ε_n^{max} adaptively during training by reducing it when a selection is rejected. This also makes the model less sensitive to initial setting of ε_n^{max}, in that we may set it to a larger value initially, and then adaptively reduce it during training as necessary.

If the learning algorithm reaches a local minimum with $\varepsilon_P^{max} < \varepsilon_n$, then either the learning rule needs to continue its search within the current weight space in order to obtain a

better optimum, or member network complexity of the ensemble is insufficient to fit the current set of exemplars. Therefore, at this point we either restart network training or modify network complexity and continue.

Finally, it may be difficult or expensive to determine whether a given set of ensemble networks can fit a set of exemplars within the chosen tolerance. We will leave this matter to discuss in Chapter 8.

To summarise, we now present an algorithm for implementing the Maximum Inter-Quartile Range criterion that embodies each of the considerations addressed above. The general algorithm begins with a set of ensemble networks of simple complexity, and increases the complexity appropriately as the exemplars become too difficult for the networks to fit. In our experiments we always begin with a set of networks which is each composed of 1 hidden unit and initial weights are randomly generated within [-0.5, +0.5] [47]. We employ the bootstrapping resampling technique to produce training subsets for training members of the ensemble respectively. During this process, when some member network complexity cannot fit training tolerance $\varepsilon_n^{\max}$ we increase the network complexity. We use multistarts to determine when the model is having difficulty fitting the exemplars. Let R count the number of restarts, and let $R^{\max}$ be the maximum number of restarts we allow before reducing training threshold $\varepsilon_n^{\max}$. Note that we reset $R = 0$ after adding a new exemplar, since each new exemplar results in a new training set, x^n, $n = 1, 2, 3, \ldots$. The algorithm outline follows.

Initialisation:

- The ensemble is composed of several network models, each model starting with a few hidden units and random weights.
- Set the training threshold for each individual member network $\varepsilon_n^{\max}$, and desired accuracy threshold $\varepsilon_P^{\max}$. The value of maximum variance may be used as the desired accuracy threshold such that the algorithm can be extended to the unlabelled data selection problem.

- The exemplar selection will be done in the input space in which the x^P is generated randomly.
- Set restart number $R^{\max}$.
- Set the minimal distance between two x_i and x_j, $d_{\min}$.

The clarification of the algorithm program:

1. Initially a small training set x^n is randomly chosen from the x^P, consisting of 10 data points, and a number of smaller subsets of data points are produced by using bootstrapping technique, which will be used to train network models of the ensemble.

2. Continue training until the desired accuracy $\varepsilon_P(x^n)$ is below down the set threshold $\varepsilon_P^{\max}$ and take the actions listed, in sequence:

(a) Train each network model, and return ε_n;

- Check whether $\varepsilon_n \leq \varepsilon_n^{\max}$. If the network model is not trained to convergence, increase the network complexity;
- Otherwise, continue training other network models;

(b) On finishing all models' training, compute inter-quartile range (x) for each x point;

- Select x_{n+1} = arg max inter-quartile range (x|x^P).
  Check whether x_{n+1} is amongst or too close to previous exemplars. If x_{n+1} is rejected, check restart value. If R< $R^{\max}$, increment R, and continue training; Otherwise, reduce the training threshold $\varepsilon_n^{\max}$, set R = 0 and continue training.
- Otherwise, append x_{n+1} to the training set x^n, set R = 0, and continue training.

6.3 Demonstrating the Technique

In the above sections we have discussed the method for actively selecting training exemplars and presented the algorithm. Here we wish to convey the qualitative behaviour of the Maximum Inter-Quartile Range criterion. We illustrate this method to obtain the target function g(x) = sin(x) illustrated in Figure 6.3-1. The ensemble is composed of 10 feedforward networks, each with one hidden layer, and initial complexity of 1 hidden unit. Each network unit employs for its transfer function the usual sigmoid output activation $o(a) = (1 + \exp(-a))/(1 - \exp(-a))$, where a is the weighted sum of the inputs to the unit. The values of weights are initially generated randomly in the range [-0.5, +0.5].

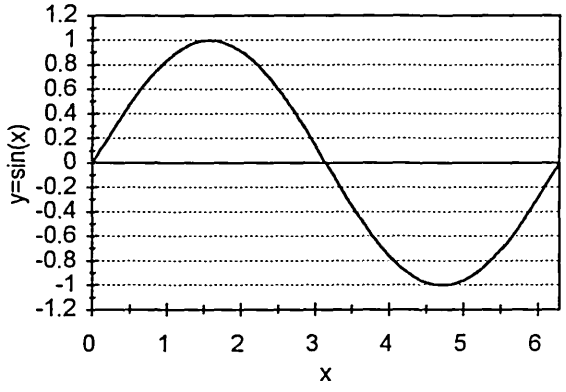


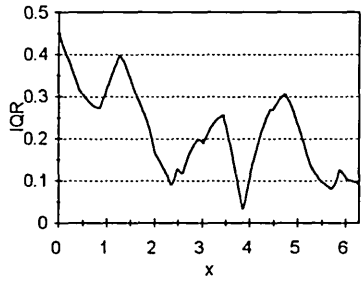
Figure 6.3-1: Target function sin(x)

Given the input X_i and values of weights W_i, correspondingly, we have different outputs of networks $Y_i = f_i(X_i, W_i)$, here X_i and W_i are vectors of input patterns and weights, i = 1, 2, 3, ..., 10. According to our definition in Section 6.2, the "inter-quartile range" is equal to the distance of $(Y_8^* - Y_3^*)$. The ensemble output is the average of the outputs of the 10 networks.

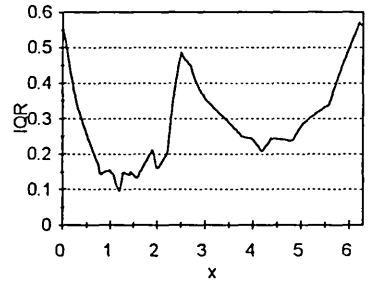
6.3.1 Training Network Ensemble on Actively Selected Exemplars

Now we can investigate what sequence of examples will be chosen by the criterion while learning the mapping of Figure 6.3-1. Examples will be chosen from a set of candidates, x^P, here P = 300, randomly generated from the input space. We followed the algorithm

given above in Section 6.2. We began with an initial training set consisting of 10 examples randomly chosen from x^P. This initial training set consists of {1.8703, 2.5637, 5.3586, 4.0347, 5.9049, 5.9260, 1.2398, 2.7739, 3.9717, 0.8826 }. Then we use the bootstrapping technique produce 10 smaller subsets by choosing half of the training set for training each of the 10 networks. The training threshold, $\varepsilon_n^{\max}$, for each network is set at a value of 0.01, and the desired accuracy threshold (stop training), $\varepsilon_P^{\max}$, for the ensemble, is 0.01. Between selection of successive examples, each member network function of the ensemble is optimised using a backpropogation gradient algorithm over the training set, x^n. Subsequent to optimisation over x^n, updating weights and obtaining w_n, the inter-quartile range(x) is calculated for each x $\in$ x^P, and x_{n+1} chosen corresponding to the largest inter-quartile range(x|x^P). The training set is updated, obtaining x^{n+1} = { x^n, x_{n+1} }. Figure 6.3.1-1 illustrates that the criterion favours candidate examples lying in regions where more information is necessary. After 8 new examples are selected, the training set is sufficient to fit all member networks of the ensemble well and the training procedure is concluded.

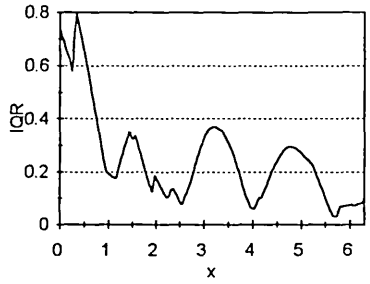


(a) n =1, x = 0.000000 selected

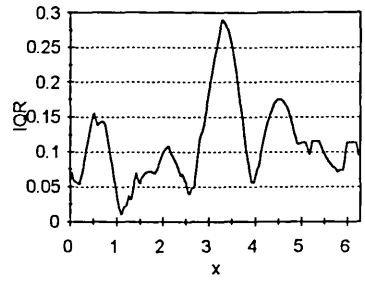


(b) n = 2, x = 6.199144 selected

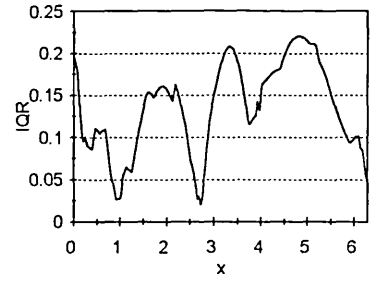
(c) n = 3, x = 6.283200 selected



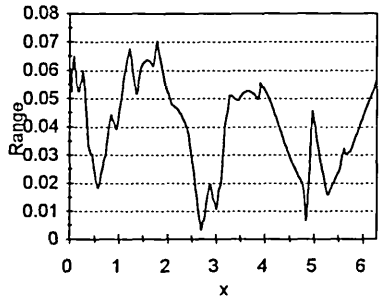
(d) n = 4, x = 0.336225 selected



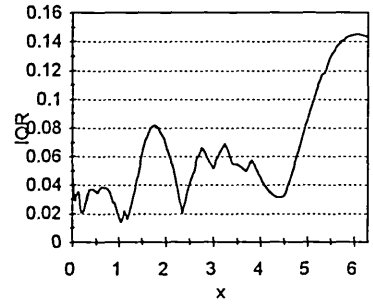
(e) n = 5, x = 3.299205 selected



(f) n = 6, x = 4.812217 selected



(g) n = 7, x = 1.786194 selected

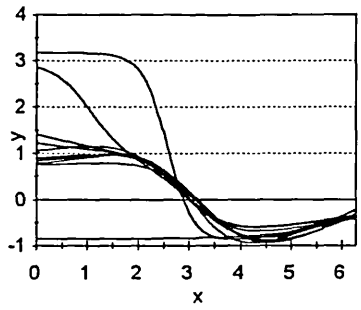


(h) n = 8, x = 6.052045 selected

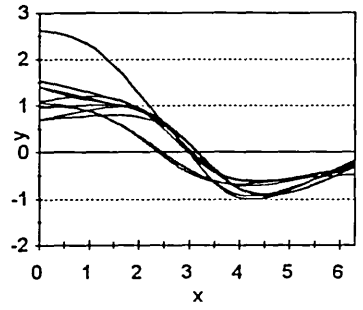
Figure 6.3.1-1: These figures illustrate the sequential selection of exemplars using MIQR active learning. For each selection, the new exemplar x_{n+1} is given by the valued x corresponding to the largest inter-quartile range($x|x^P$).

6.3.2 Illustrating the Behaviour of Outputs of Networks

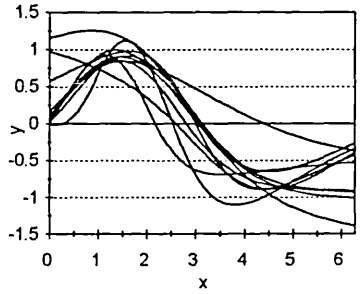
As the training set is growing, member networks of ensemble have increasing probability of being trained by more and more particular information. Thus they are optimised gradually. If the models agree closely then the training set contains sufficient information for the learning task. If the models disagree then additional exemplars need to be collected and this selection procedure will be repeated. The model disparity influences the collection of future exemplars. This process is iterated until the models agree sufficiently. Figure 6.3.2-1 illustrates the evolution of ensemble model during learning from beginning to end.



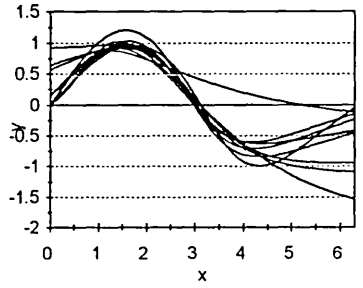
(a) n = 0 (initial 10 models)



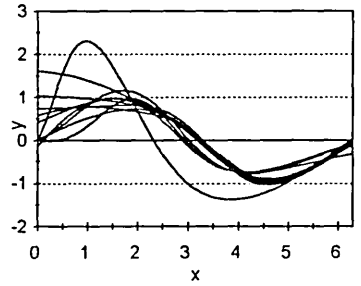
(b) n = 1



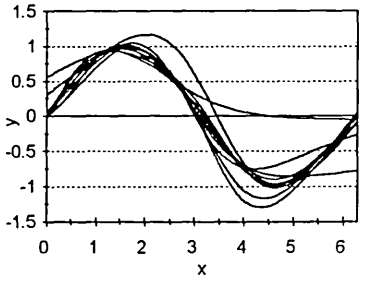
(c) n = 2



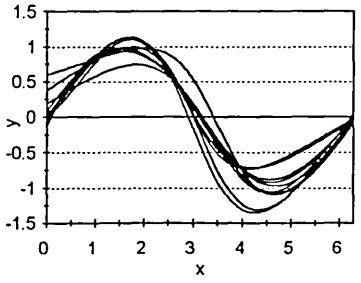
(d) n = 3



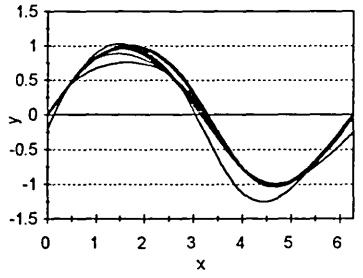
(e) n = 4



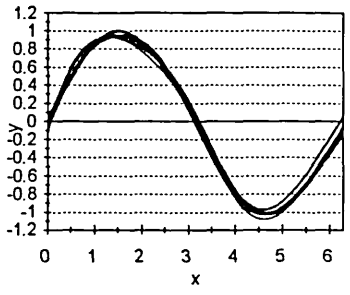
(f) n = 5



(g) n = 6



(h) n = 7



(i) n = 8 (Final agreeing output)

Figure 6.3.2-1: The evolution of the outputs of the ensemble from beginning to end.

6.3.3 The Relationship between Data Selection and Training Threshold

We now illustrate how the selection criterion is related to the training threshold and network complexity. Experience indicates that if the training threshold is too large or too loose, the network model is unable to fit the learning task well because of insufficient information. On the other hand, if it is too small or too strict, learning will be more costly. In order to achieve the desired learning level and not consume too much cost, we need to be allowed to adapt the training threshold and grow the network complexity as necessary. We initialise each member network of the ensemble with 1 hidden unit, and set the training threshold $\varepsilon_n = 0.01$. As exemplar selection proceeds, the algorithm automatically modifies network complexity by adding another hidden unit when the network can no longer fit the exemplars. In addition, we set the maximum restart $R^{max} = 3$ so that when the model repeatedly selects exemplars amongst or close to the previous training exemplars the model will automatically reduce the training threshold by 30%. When the model repeatedly selects the same examples as before, this means the network complexity is too simple to fit the learning task, since it is unable to discover further new information. During learning, the training threshold will be becoming more and more strict, subsequently, the network complexity will become bigger and bigger until it fits the learning task well enough. Table 6.3.3-1 illustrates this relationship.

From Table 6.3.3-1, we can see that in the beginning, the training threshold $\varepsilon_n^{\max}$ keeps the same value as the initial 0.01 while four exemplar selections have been made. From the fifth exemplar selection, it drops down considerably, being reduced from 0.01 to 0.002401. This means that the training threshold is becoming so loose that it is very difficult for the model to discover a new exemplar. When active selection is proceeding, the training threshold value continues to be reduced till it reaches 0.000824 when the selection procedure ceases.

At the same time, we look at the evolution of the network complexity according to Table 6.3.3-2. As the training threshold $\varepsilon_n^{\max}$ is becoming more strict and the training set is becoming bigger and bigger, the network complexity is also increasing, but very slowly compared to the rate of the training threshold while the training set is growing. Clearly, the complexity of one hidden unit is insufficient to fit the learning, so all members of the ensemble increase 1 hidden unit after the first iteration although the training threshold remains the same with a value of 0.01. However, from the second iteration, all members of the ensemble keep the same complexity until the training ceases even when the training threshold has become very small. This situation means that for sin(x) the network complexity of 2 hidden units has been sufficient to fit the learning.

Table 6.3.3-1 Data Selection for sin(x)

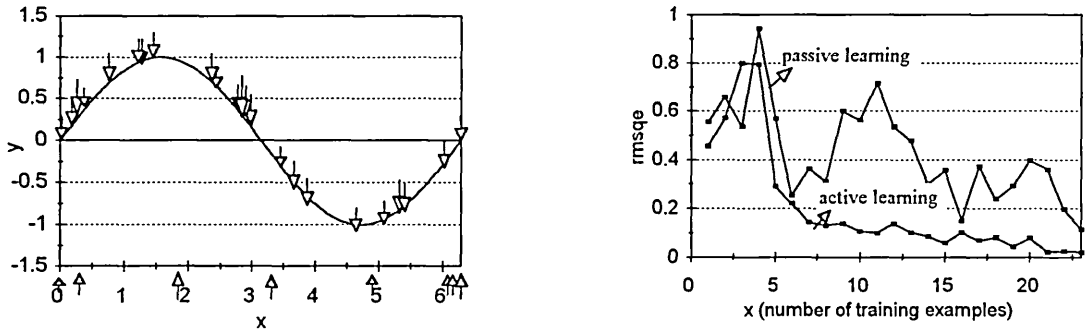
Iteration	Largest inter-quartile range	Data selected	Training threshold	Maximum variance	ε_P (rmse)
1	0.456044	0.000000	0.01	0.2855	0.4535
2	0.569854	6.199144	0.01	0.1631	0.5810
3	0.799543	6.283200	0.01	0.2157	0.8011
4	0.794538	0.336225	0.01	0.2973	0.7964
5	0.290039	3.299205	0.002401	0.1096	0.2487
6	0.220716	4.812217	0.002401	0.0842	0.2108
7	0.070152	1.786194	0.001176	0.0485	0.1705
8	0.145097	6.052045	0.000824	0.0056	0.1519

Table 6.3.3-2 Network Complexity for sin(x)

Iteration	Net1 units	Net2 units	Net3 units	Net4 units	Net5 units	Net6 units	Net7 units	Net8 units	Net9 units	Net10 units
0	1	1	1	1	1	1	1	1	1	1
1	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2
3	2	2	2	2	2	2	2	2	2	2
4	2	2	2	2	2	2	2	2	2	2
5	2	2	2	2	2	2	2	2	2	2
6	2	2	2	2	2	2	2	2	2	2
7	2	2	2	2	2	2	2	2	2	2
8	2	2	2	2	2	2	2	2	2	2

6.4 Data Selection and Generalisation Performance; Comparison with Passive Learning

The motive of active learning is to minimise the amount of data selection and maximise generalisation performance. We now examine whether our method is advantageous over traditional methods, for instance, random selection, or passive learning. We do the experiments again, but this time we select examples randomly instead of selecting training examples corresponding the largest inter-quartile range.



(a) Data selection

(b) Generalisation performance

Figure 6.4-1: Data selection and generalisation comparison of MIQR and passive learning
↑ indicates MIQR data selection and ↓ Passive data selection.

The method for randomly selecting training examples still converges but obviously, many more exemplars are required to meet the desired learning level using the passive selection method. Furthermore, we compare the generalisation performance of these two distinct learning schemes (active learning and passive learning). We can see the generalisation (measured by root mean squared error) of active learning outperforms that of passive learning. The results are indicated in Figure 6.4-1.

6.5 Discussion

We have proposed and discussed the practical application of our active exemplar selection method. We emphasise the importance of obtaining a good solution to the training problem at each stage. We experimented briefly with an iterative training regime, in which little optimisation was performed between successive examples. We found that although our method for selecting exemplars outweighs a contending method (Maximum Variance, discussed in the next chapter), there was still a tendency to select repetitively near previously selected data points when the optimisation between successive examples was not stringent. In some such cases a "pathological" phenomenon happened in which the algorithm entered a cycle, endlessly alternating between two examples. For this reason, we advocate stringent optimisation at each step by means of automatically reducing the training threshold as necessary.

In previous work, although researchers have employed the ensemble method, they almost invariably use a fixed architecture of networks. We noted in practice that a fixed architecture of networks is very difficult to regulate to fit the learning task, when, in particular, an ensemble of networks is used to approach a complicated mapping, for instance, to approach a discontinuous function. We will discuss this issue in more depth in the next chapter. Furthermore, using a flexible architecture of networks, we therefore automatically create an ensemble, whose members have different numbers of hidden units.

Recall that we discussed in Chapters 4 and 5 that we require members of the ensemble as different as possible, thus we have a higher probability that members of the ensemble are relatively independent in functional space, which guarantees that the variance of the final ensemble output is $1/N$ of the variance of the individual output.

The advantages of active exemplar selection are apparent both in data selection and generalisation performance. As for the overhead due to the selection process, Plutowski has made a detailed investigation, from which he concluded that the overhead of using an active selection method, including computation, running time, selection process, is reduced more than that of using other passive selection methods in general, such as an evenly-spaced method, or a random sampling method. Note that Plutowski uses the Δ ISB method, which needs to compute the Hessian matrix. Since our method only uses the very simple computation of maximum inter-quartile range, it is much cheaper than the Δ ISB method in computation.

One issue must be addressed further. There are two important criteria in an active learning process, one is to decide how to select the next training exemplar, such as MIQR (our newly proposed method), Maximum Variance, and Δ ISB etc. The another is to decide when to stop training, namely, by what can we conclude the network models have reached a desired accuracy of training? In section 6.2.2, we specify a tolerance $\varepsilon_p^{\max}$ representing the desired accuracy measured by $\varepsilon_P = \left[\frac{1}{P} \sum_{x \in x^P} (g(x) - f_{ens}(x))^2 \right]^{1/2}$. However, this measure is applicable to only deterministic mappings, *i.e.* a "labelled" data problem. More generally, we want our method to be applicable to "unlabelled" data problems. That is to say, without knowing output values of target function $g(x)$ corresponding to input space, we are able to judge if the networks have been trained to fit the mappings well enough to the desired accuracy. If we are indeed to do so, the benefit is outstanding. Firstly, we can apply our method to unknown mappings, in which we need to be provided by only a small number of examples. Secondly, we can easily extend our method to noisy environmental data selection. Thus our method is applicable to more general problems. Then, what is

unlabelled data in our computations? Obviously, variance and inter-quartile ranges of the output of ensemble networks are unlabelled data because they are only dependent upon the computational results of ensemble networks' outputs, having nothing to do with the target output. We suggest the use of the variance measure as a criterion for stopping training, instead of the inter-quartile range measure. The reason for us to do so is very easily understood. Recall that the inter-quartile range is defined as the difference between upper quartile and lower quartile. By this token, we cannot be sure that all outputs are "agreeing" according to maximum inter-quartile range dropping below a set value of the threshold since the inter-quartile range does not include outliers of the outputs of the ensemble. On the other hand, a variance measure truly indicates whether or not all networks of the ensemble agree. It seems to be a little perplexing that we use the inter-quartile range as a criterion for selecting exemplars while we use variance as a criterion for judging when to stop training. Note that we have ingeniously combined the advantages of both inter-quartile range and variance into our method.

Chapter 7

Experience with Exemplar Selection on a Continuous Function and a Discontinuous Function

7.1 Summary

In Chapter 6, we developed an *active exemplar selection* method called the Maximum Inter-Quartile Range (MIQR) criterion. The method is *active* in that it utilises the ensemble network state during learning. Given a set of networks and a fixed set of available examples generated randomly, the method selects a concise subset for training. The exemplar selection procedure is dependent upon the general tendency of the output of ensemble networks after ignoring outliers. In this sense, this method is implemented based on a very simple, but most important ancient philosophy: ***the minority obeys the majority***. Fitting the selected exemplars results in the entire set being fitted as well as desired. The algorithm also incorporates a method for regulating the network's complexity, automatically adding exemplars and hidden units as needed.

Here, we present empirical results from applying the technique to approximate two distinctive functions: one is a continuous function (cos(x)), and the another is a discontinuous function (the square wave). We intentionally introduce these two distinctive functions in order to observe how well our method works for a discontinuous function approximation, in which much previous work appears to be weak. The results indicate that the MIQR obtains smaller data sets (average 29.2 for the discontinuous

function and 7.8 for the continuous function), whereas passive learning requires much more data selection for the same learning tasks (average 65.4 for the discontinuous function and 19 for the continuous function). There is also clear evidence of improved generalisation performance.

More importantly, in addition to the comparison with passive learning, we compare in detail the MIQR method with the Maximum Variance exemplar selection technique, which is widely used by many researchers[59][89][75][96][35] and is an heuristic similar to MIQR that also qualifies as active exemplar selection. Although the two active selection methods performed best overall, our MIQR is more effective than Maximum Variance, in particular when approaching a discontinuous function.

It is clear that it is much more difficult to approach a discontinuous function than to approach a continuous function.

7.2 Introduction

In Chapter 6 we developed a new method of *active selection* of training exemplars for neural network learning. Active selection uses information about the state of the ensemble networks when choosing new exemplars. The approach integrates the bootstrapping technique, the network ensemble method and a statistical sampling method to derive a "greedy" selection method picking the training example that maximises the density of information by "ignoring" outliers. It should be noted that after outliers are ignored, the output trend of the entire ensemble model is guaranteed to agree in general. This phenomenon reflects a general law: the minority obeys the majority. We refer to this method as Maximum Inter-Quartile Range (MIQR). Besides, the method automatically regulates network complexity by adding to the hidden units as necessary to fit the selected exemplars, and terminates when the ensemble model fits the entire set of available examples to the desired accuracy. Hence the method is a nonparametric regression technique. In this chapter we explore the possible benefits of active exemplar selection by comparing the method with traditional methods (including passive learning and other contending active learning), on both the continuous function and discontinuous

function. Note that theoretical and practical facts indicate that the effectiveness of methods for selecting exemplars is principally dependent upon the learning task. So far existing methods for active learning are weak when approaching discontinuous functions. Thus experimental comparison on these two distinctive functions is significant. These results indicate that when approaching a discontinuous function, the methods for selecting exemplars will decrease the effectiveness to some extent, compared with when approaching a continuous function. No matter whether active learning or passive learning a complicated learning task will consume more than a simple learning task (here a continuous function). The comparison shows MIQR is effective.

Having demonstrated that this particular type of exemplar selection is worthwhile, we compare MIQR with Maximum Variance exemplar selection in detail. We compare the total number of selected exemplars and generalisation performance, as well as total number of iterations of training. We find that although two active selection methods select the most concise training sets, and provide an improved generalisation performance, MIQR is more efficient than Maximum Variance, in particular, when approaching a discontinuous function.

7.3 The Algorithm

Before formal experiments are conducted, it is necessary to recall the algorithm for our Maximum Inter-Quartile Range method for actively selecting training exemplars.

Initialisation:

- The ensemble is composed of 10 network models, each model starting with 1 hidden unit and random weights within the range of [-0.5, +0.5].

- The training threshold for each individual member network $\varepsilon_n^{\max}$ is initially set to 0.01; of course, it will probably become a smaller value as the training proceeds further. The desired accuracy threshold $\varepsilon_p^{\text{var}}$ is fixed at 0.01. Here we use *variance*

(instead of *root mean squared error* ε_p which was defined in Chapter 6) as the "stop-training" criterion such that the algorithm can be extended to apply to the unlabelled data selection problem. Although intuitively the desired accuracy $\varepsilon_p^{\text{var}}$ can be set more stringently to approach a continuous function than a discontinuous function, it is set at the same value for both the functions in question in order for us to observed the experimental results based on the same initialisation conditions.

- The exemplar selection will be done in the input space in which x^P consists of 300 points which are generated randomly.

- Initially a small training set x^n is randomly chosen from the x^P, consisting of 10 data points, and 10 smaller subsets of data points are produced by using a bootstrapping technique, which will be used to train 10 network models.

- The maximum number of restarts, $R^{\max} = 3$.

- The minimal distance between two x_i and x_j, $d_{\min} = | x_i - x_j | = 0.02$.

The Algorithm

$R = 0$.

Do

{

Train 10 networks.

If ($\varepsilon_n > \varepsilon_n^{\max}$)

{

Add one hidden unit to the network.

}

Compute inter-quartile range (x);

Select a new exemplar, x_{n+1} = arg max inter-quartile range (x | x^P).

If ($|x_{n+1} - x_i| \leq d_{\min}$) for some *i* in { 0, 1, 2, ..., n}.

{

Reject x_{n+1}, and set $R = R+1$;

If $(R \geq R^{\max})$

{

Reducing the training threshold $\varepsilon_n^{\max}$ by 30%.

Set R =0, and ***Break***;

}

}

Else // if x_{n+1} is not amongst or close to previous exemplars;

{

Append x_{n+1} to previous training set x^n, and increment n.

set R = 0, and ***Break***;

}

} **While** $(\varepsilon_n(x^P) > \varepsilon_p^{\text{var}})$

7.4 Experiments with Neural Networks

Previous work has been done on the feasibility of an active learning method by many researchers[89][88] [59] [112] [22] [100]. We now apply the MIQR method for active selection to practical experiments. In particular, we lay stress on the comparison of results of their application to two distinctive function approximations, here, the *square wave* function and *cos*(*x*) (see Figure 7.4-1).

We intentionally introduce the square wave (a discontinuous function) and cos(x) (a continuous function) in order that we can observe the effects of the method being applied to both discontinuous and continuous functions. For each function approximation, training data were obtained by the bootstrapping technique, i.e. by subsampling randomly from an initial small random data sample collected from the same X-axis range of [0, 2π] and the corresponding $g(x)$.

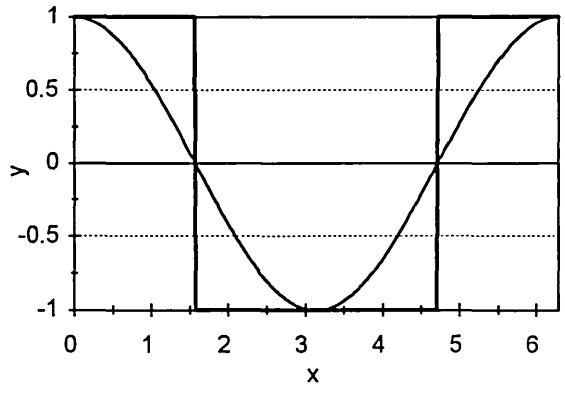
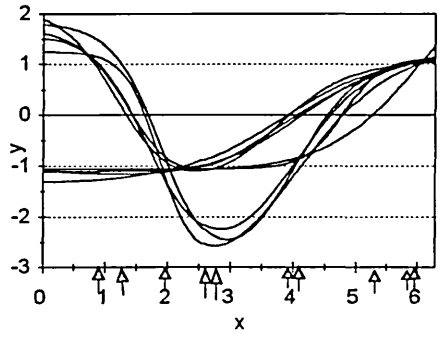


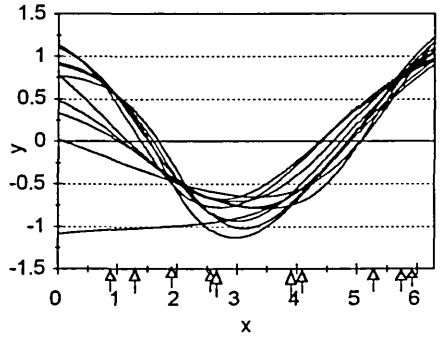
Figure 7.4-1: $g(x) = \cos(x)$ and $g(x) = \begin{cases} 1, & 0 \le x < \pi/2 \\ -1, & \pi/2 \le x < 3\pi/2; \\ 1, & 3\pi/2 \le x \le 2\pi. \end{cases}$

The backpropagation algorithm was used to train the neural network models. The bootstrap technique is a very effective tool in ensemble network learning and has been extensively studied by many researchers [80][89][104][90][82]etc. The training set initially consisted of 10 points (input-output pairs) [89] chosen randomly from a fixed set of 300 points, which are generated randomly from the input space [0, 2π]. A subsample set (also selected randomly) consisting of half of the training set was applied to train one member network of the ensemble. Another 9 such random subsample sets of equal size were subsequently used to train the other 9 member networks respectively. It is not surprising that the 10 network models disagree at the initial stage (Figure 7.4-2).



square wave

Figure 7.4-2(a): Output from the 10 models after the first training iteration on initial data sets. ↑ indicates initial points.



cos(x)

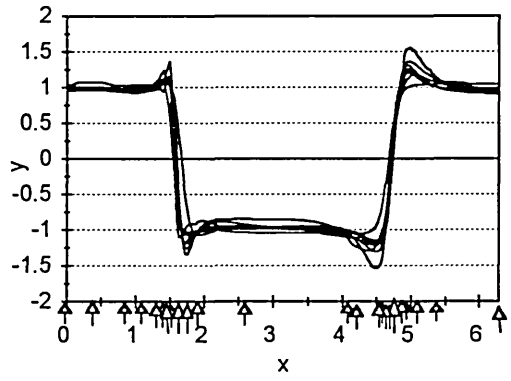
Figure 7.4-2(b): Output from the 10 models after the first training iteration on initial data sets. ↑ indicates initial points.

The inter-quartile range of the models was then computed at the 300 randomly chosen points (the initially generated ones). Of course, the points of higher inter-quartile range are the points where more information is needed - the querying criterion for resampling. If we denote x_i to be 300 randomly chosen data points from [0, 2π], then the outputs of 10 models are $y_N(x_i)$, i = 1, 2, 3, ..., 300; N = 1, 2, 3, ..., 10. For each i we rearrange them to ascending order, namely,

$$Y_1^*(x_i) \le Y_2^*(x_i) \le Y_3^*(x_i) \le \ , \ldots , \le \ , Y_9^*(x_i) \le Y_{10}^*(x_i)$$

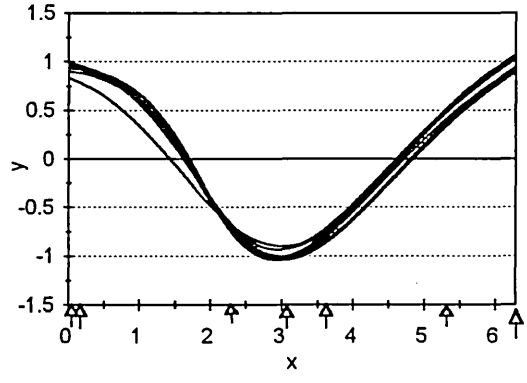
Thus, the Inter-Quartile Range (x_i) of the 10 models is computed as follows:

$$\text{Inter-Quartile Range}(x_i) = Y_8^*(x_i) - Y_3^*(x_i)$$



↑ resampled points

Figure 7.4-3(a): 10 final "agreeing" models for the square wave after 27 resampled data points.



↑ resampled points

Figure 7.4-3(b): 10 final agreeing models for *cos(x)* after 7 resampled data points.

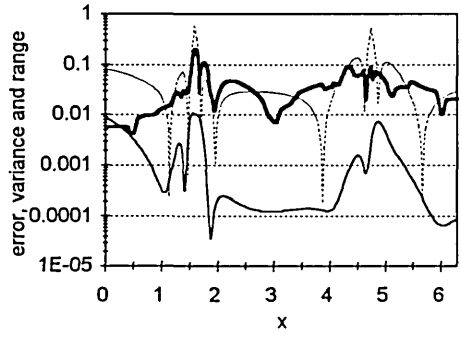
The point corresponding to the largest Inter-Quartile Range value was then resampled and added to the training set. Again 10 sets of random subsamples of half the data were used to train the 10 neural networks. The models were recompared by computing the inter-quartile range as before. As long as the maximum variance remained above a desired accuracy value of 0.01 the process of resampling the additional point of the largest inter-quartile range was repeated, added to the earlier sampled data and the networks were retrained with half the updated sample each time. When the maximum variance dropped below the set threshold the iterations ceased. Figure 7.4-3 indicates how closely the models agreed with one another. If we use $g(x)$ to denote the target function, then,

$$y_{ensemble}(x_i) = \overline{y(x_i)} = \frac{1}{10}\sum_{m=1}^{10} y_m(x_i) \tag{7.1}$$

$$\text{Var}(x_i) = \frac{1}{10}\sum_{m=1}^{10} (\overline{y(x_i)} - y_m(x_i))^2 \tag{7.2}$$

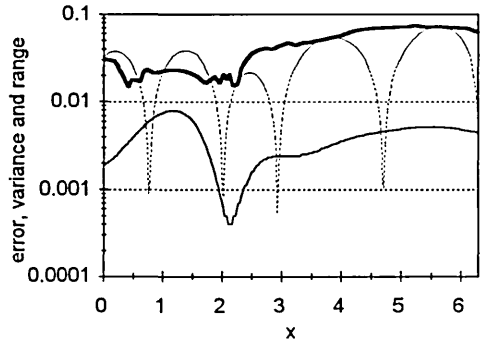
$$\text{Error}(x_i) = (y_{ensemble}(x_i) - g(x_i))^2 \tag{7.3}$$

Here, $y_{ensemble}$ is the final output obtained by averaging the outputs of the 10 network models, y_m is the individual output of a member network model of the ensemble, Var is the variance, and Error is the squared error between ensemble output and the target. i = 1, 2, 3..., 300. The final variance, square error and inter-quartile range plots are shown in Figure 7.4-4, from which it is seen that the for the discontinuous function, the peaks of error, variance and inter-quartile range are more distinguished near the discontinuities. In contrast, the continuous function has no peak because there is no "abruptness" in the continuous function.



square wave

Figure 7.4-4(a): Variance (thin line), Square Error (broken line) and Inter-Quartile Range (thick line) plots of the 10 final models at 300 points randomly selected from the same X-axis range of [0, 2 π].



cos(x)

Figure 7.4-4(b): Variance (thin line), Square Error (broken line) and Inter-Quartile Range (thick line) plots of the 10 final models at 300 points randomly selected from the same X-axis range of [0, 2 π].

The initial 10 random data samples were: {1.8703, 2.5637, 5.3586, 4.0347, 5.9049, 5.9260, 1.2398, 2.7739, 3.9717, 0.8826}. The resampled data were collected for the *square wave* and *cos(x)* approximations respectively in the following order (see Table 7.4-1). Figure 7.4-5 indicates the ensemble output and the resampled data point distribution in the input space (including initial points).

Table 7.4-1: active data selection for the square wave and cos(x)

Order of resampled data	Square wave	cos(x)
1	0.483323	0.273183
2	1.807208	3.614416
3	5.337568	2.206475
4	0.000000	6.283200
5	2.605742	3.089065
6	4.076725	0.000000
7	5.127428	5.232498
8	1.618082	
9	6.283200	
10	1.260843	
11	0.945632	
12	1.386927	
13	1.639096	
14	1.134758	
15	1.428955	
16	4.139767	
17	4.812217	
18	4.644104	
19	4.623090	
20	4.854245	
21	1.534025	
22	4.833231	
23	1.513011	
24	4.560048	
25	1.660110	
26	4.602076	
27	1.597068	

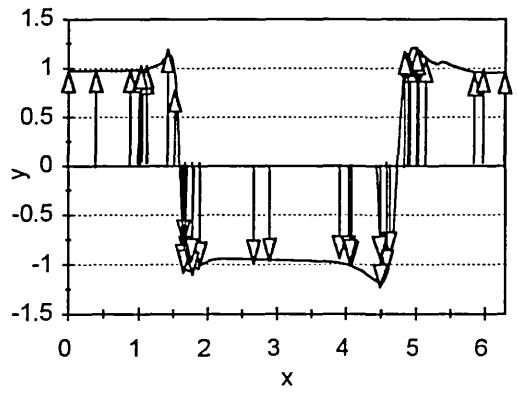


Figure 7.4-5(a): Final ensemble output and resampled data distribution (including initial data) by MIQR for the square wave

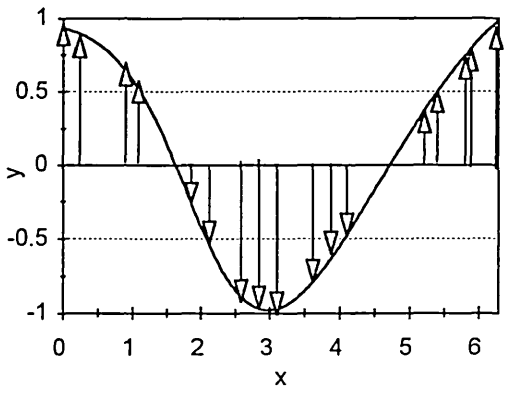


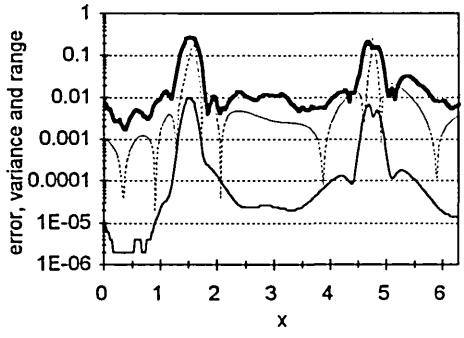
Figure 7.4-5(b): Final ensemble output and resampled data distribution (including initial data) by MIQR for cos(x)

7.5 Comparison with Passive Learning for Data Selection

We did the experiments again and on this occasion, we added a point randomly (passive learning) instead of adding the point corresponding to the largest inter-quartile range

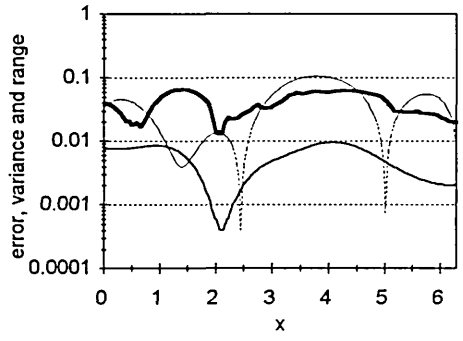
value. The algorithm also converged, but as expected, more data points were collected than in the experiment with active learning. It needs 65 data points to approximate the square wave and 23 data points to approximate cos(x). The final variance and square error plots are shown in Figure 7.5-1.

From the above experiments, it seems to be more "difficult" to approximate the *square wave* function than *cos(x)*. Resampled data points needed in passive learning are more than in active learning. Besides, in order for convergence, it needed 200000 ~ 300000 cycles to approximate the *square wave*, whereas it needed only 50000 ~ 60000 cycles to approximate *cos(x)*. In order to furtherly study this interesting phenomenon, we continue to do additional experiments with different initial data sets.



square wave

Figure 7.5-1(a): Variance (thin line), Square Error (broken line) and Inter-Quartile Range (thick line) plots of the 10 final models by passive learning at 300 points randomly selected from the same X-axis range of [0, 2 π].



cos(x)

Figure 7.5-1(b): Variance (thin line), Square Error (broken line) and Inter-Quartile Range (thick line) plots of the 10 final models by passive learning at 300 points randomly selected from the same X-axis range of [0, 2 π].

7.6 Further Experiments

Because it is not sufficient evidence doing only one experiment to demonstrate that the active learning outweighs the passive learning, other experiments need to be done to confirm this. We repeated the experiments 4 times with different initial random seeds so as to generate different environmental input spaces as discussed in Chapter 2. We

observe the results of active learning taking place in different environmental input spaces to justify the new method for active exemplar selection.

7.6.1 Data Selection Comparison

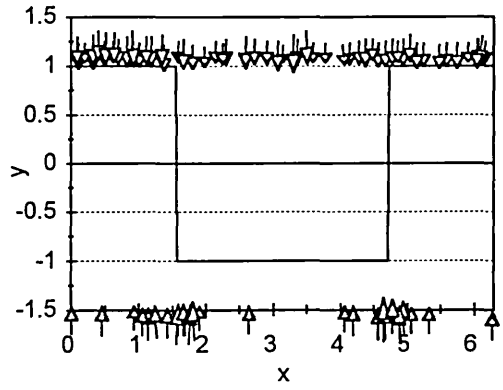
The experimental results (including the first) are presented in Table 7.6.1-1, from which it is obvious that the number of resampled data points for the continuous function (cos(x)) is much less than that for the discontinuous function (the square wave), whether by active learning or by passive learning. In the case of active learning, the amount of data resampled averages 29.2 points for the square wave, and 7.8 points for cos(x), whereas the amount of data resampled averages 65.4 points for the square wave, and 23.2 points for cos(x) in the case of passive learning. In addition, more importantly, the distribution of data resampled is very interesting, especially in the case of approximation for the discontinuous function (see Table 7.6.1-2, Figure 7.6.1-1 and Figure 7.6.1-2). There is a sampling trend which is noteworthy here. Unlike the passive learner, the active learner does not simply sample the domain on a uniform grid. Instead it chooses to cluster its data points typically around the "discontinuity" or "uncertainty" location. This phenomenon is more noticeable when approximating the discontinuous function.

Table 7.6.1-1 Amount of Data Resampled

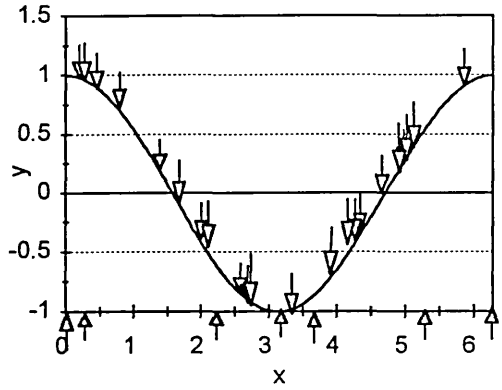
	MIQR *(square wave)*	Passive *(square wave)*	MIQR *cos(x)*	Passive *cos(x)*
Experiment 1	27	65	7	21
Experiment 2	33	61	8	17
Experiment 3	31	70	7	24
Experiment 4	29	64	7	14
Experiment 5	28	67	10	19
Total points	146	327	39	95
Average points	29.2	65.4	7.8	19

Table 7.6.1-2 Distribution of Data Resampled in the Input Space

X-axis [0, 2 π]	MIQR *(square wave)*	Passive *(square wave)*	MIQR *cos(x)*	Passive *cos(x)*
0-1	16	58	10	15
1-2	54	51	4	20
2-3	2	42	2	14
3-4	1	52	6	9
4-5	51	47	8	16
5-6	12	40	4	13
6-2 π	10	37	5	8



↑ — Points resampled by active learning
↓— Points resampled by passive learning
Figure 7.6.1-1(a): The resampled data points for the *square wave*



↑ — Points resampled by active learning
↓— Points resampled by passive learning
Figure 7.6.1-1(b): The resampled data points for *cos(x)*

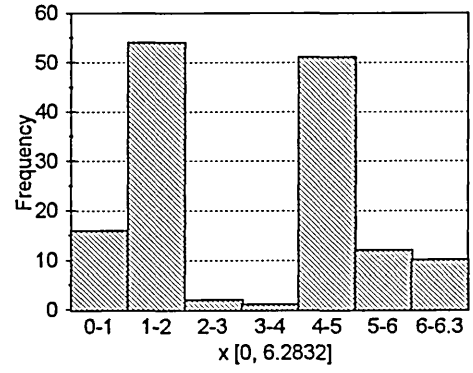


Figure 7.6.1-2(a): Frequency distribution of data points resampled by MIQR active learning for the *square wave*.

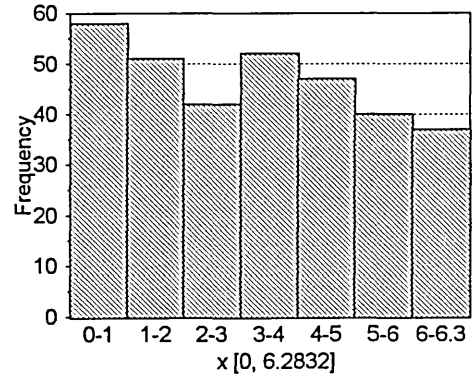


Figure 7.6.1-2(b): Frequency distribution of data points resampled by passive learning for the *square wave*.

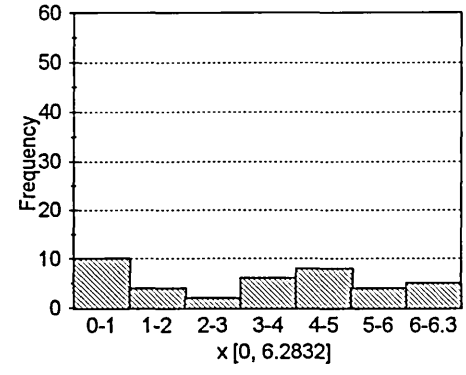


Figure 7.6.1-2(c): Frequency distribution of data points resampled by MIQR active learning for *cos(x)*.

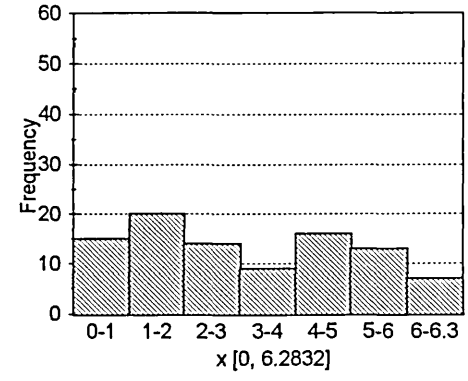
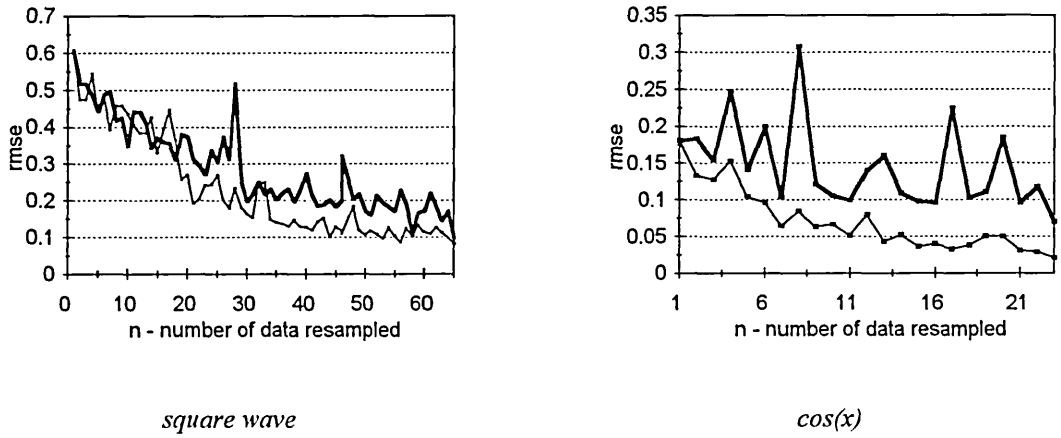


Figure 7.6.1-2(d): Frequency distribution of data points resampled by passive learning for *cos(x)*.

7.6.2 Generalisation Comparison

Now we examine generalisation performance. Generalisation is the ability of a neural network to correctly classify new patterns. Most often, generalisation is measured by the mean squared error or root mean squared error in the entire input space. In this experiment, the input space is fixed at the range of $[0, 2\pi]$, from which 300 examples are chosen randomly. Figure 7.6.2-1 shows generalisation performance via number of data resampled. Passive learning curves are more oscillatory than active learning curves no matter whether the continuous function or the discontinuous function, since active learning judiciously selects examples whereas passive learning randomly picks examples. Therefore active learning can improve generalisation performance.



square wave *cos(x)*

Figure 7.6.2-1: As more data points are resampled, the root mean squared error declines gradually. There is also clear evidence that the generalisation performance using MIQR active learning is better than using passive learning. The thick line is the passive learning curve and thin line is the MIQR active learning curve.

7.6.3 Network Complexity Comparison

RayChaudhuri uses a set of networks with 3 fixed hidden units to approximate a continuous function [89] while Krogh uses a fixed set of networks with 20 hidden units to approximate a discontinuous function [59]. In our algorithm, the network architecture is flexible and will be growing as necessary, but all networks start with only 1 hidden unit. Recent theoretical results suggest a close relationship between the size of a network and the number of exemplars required for good generalisation[7][69]. According to this

theory, the ratio of exemplars to number of weights stands for the efficiency of a method for selecting exemplars, i.e. the lower the ratio, the more efficient the method for selecting exemplars. Now let us see if the method developed in this dissertation is more "efficient" than traditional passive learning.

Table 7.6.3-1 The Ratio of Examples to Weights

Networks	MIQR for the square wave (hidden units h)	Passive for the square wave (hidden units h)	MIQR for cos(x) (hidden units h)	Passive for cos(x) (hidden units h)
1	6	7	2	2
2	7	6	2	2
3	6	5	2	2
4	8	4	2	2
5	5	6	2	2
6	7	5	2	2
7	7	5	2	2
8	6	5	2	2
9	8	5	2	2
10	7	7	2	2
Average hidden units h	6.1	5.5	2	2
Number of weights $w = 3 \times h+1$	19.3	17.5	7	7
Number of data resampled	29.2	65.4	7.8	19
Ratio of data resampled to number of weight	1.51	3.75	1.11	2.71

Ratio of data resampled to number of weights stands for network learning efficiency using a method for data selection.

Table 7.6.3-1 gives the results for the experiments applying Maximum Inter-Quartile Range and random selection. These numbers give the ratio of the number of exemplars to the number of weights when the member networks of the ensemble were forced to grow, indicating how many exemplars can be selected before the networks become "saturated", unable to fit the selected exemplars. This indicates how informative the exemplars are. From this table, we can see that MIQR active learning has a lower ratio than passive learning, with the ratio being 1.51 for the discontinuous function and 1.11 for the continuous function. In contrast, passive learning has a higher ratio, with the ratio being 3.75 for the discontinuous function and 2.71 for the continuous function, respectively. In addition, it also corroborates the conjecture that it is more difficult to map the discontinuous function than to map the continuous function although using the

same active learning method because of the ratio being 1.51 for the discontinuous function and 1.1 for the continuous function respectively. From Table 7.6.3-1, we can see the network complexity is dependent upon the learning task, having almost nothing to do with the method for selecting examples (no matter whether active learning or passive learning).

The experimental results here are in accordance with the theoretical result that the ratio of number of exemplars selected by active learning to the number of network weights is equal to a small constant just greater than 1 [69][88][7].

7.7 Comparison with a Contending Method - Maximum Variance

The results above demonstrate clearly that the Maximum Inter-Quartile Range active learning method can cut the training data dramatically, and is also advantageous over passive learning in generalisation performance and network learning efficiency. Reviewing previous active learning methods, such as Maximum Error and Maximum Variance, it is found that Maximum Error not only is computationally complicated and expensive, but requires the assumption that the function to be approached is reasonably smooth because these Maximum Error methods have to compute the inversion of a huge Hessian matrix such as Plutowski's Δ ISB and Sung et.al's EISD (expected integrated squared difference). It is unable to approach a discontinuous function such as the square wave discussed above, because we cannot compute the Hessian matrix. In addition, Maximum Error only uses a single network, therefore it cannot make use of advantages of ensemble networks to improve generalisation. Furthermore, the Maximum Error method lacks a clear stopping criterion because it only validates labelled data [88]. Another contending method is Maximum Variance, which is very popular in ensemble network learning. However, in practice, it is found that the Maximum Variance method is frequently affected local minima and quits training too early or fails to obtain a desired training accuracy. Theoretically, exemplar selection by the Maximum Variance criterion is essentially based on the assumption that values used in the computation of variance should be normally-distributed. The frequent presence of outliers in network outputs

indicates that this is not the case. As a result, when using the Maximum Variance criterion, the training may be trapped in a "dead valley" because when approaching a particular problem, for example, a discontinuous function approximation, the learning procedure repeatedly locates the same data point and wastes a large number of useless iterations because of the disproportionate influence of outliers on the variance. Now we examine the Maximum Variance and compare it with our MIQR method.

We do the experiments again, but this time by using Maximum Variance criterion, i.e, picking up new exemplars corresponding the maximum variance instead of maximum inter-quartile range of the outputs of the member networks of the ensemble. We do the experiments starting with different random seeds so that we have different initial environmental distributions.

7.7.1 Data Selection Comparison

As demonstrated by previous work, the Maximum Variance method also is an active learning technique, it outperforms other traditional sampling techniques such as evenly-spaced and randomly sampling or passive learning techniques[89][59][88]. Now we examine this method compared with our new MIQR method.

Table 7.7.1-1 indicates the results of data selection using Maximum Variance criterion and MIQR criterion according to 5 experiments done, and Figure 7.7.1-1 indicates the data point distribution using these two criteria respectively. From Table 7.7.1-1, we can see that on average, the number of data points resampled by using MIQR is less than that

Table 7.7.1-1 Amount of Data Resampled

Experiment	MIQR (square wave)	MV (square wave)	MIQR (cos(x))	MV (cos(x))
1	27	30	7	11
2	33	38	8	8
3	31	37	7	8
4	29	35	7	10
5	28	33	10	7
Total points	146	175	39	44
Average points	29.2	35	7.8	8.8

by Maximum Variance, both for the continuous function (cos(x)) and the discontinuous function (square wave). This means data points resampled by using MIQR are more informative than those using Maximum Variance, which is corroborated by Figure 7.7.1-1, in which it seems that the data point distribution is more reasonable by using MIQR than by using Maximum Variance.

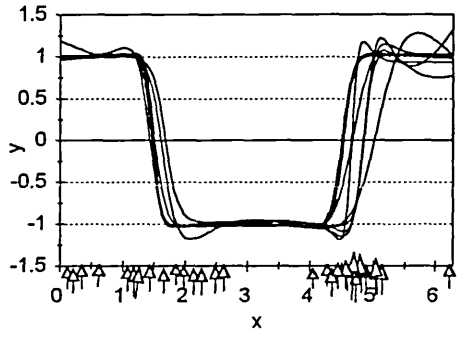


Figure 7.7.1-1(a): 10 final "agreeing" models for the square wave after 30 resampled data points using the Maximum Variance criterion

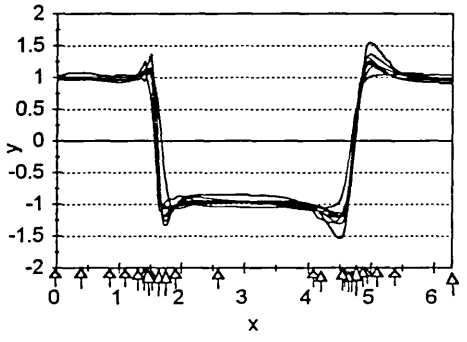


Figure 7.7.1-1(b): 10 final "agreeing" models for the square wave after 27 resampled data points using the MIQR criterion

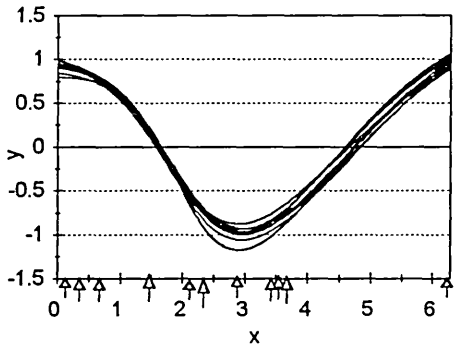


Figure 7.7.1-1(c): 10 final "agreeing" models for cos(x) after 11 resampled data points using the Maximum Variance criterion

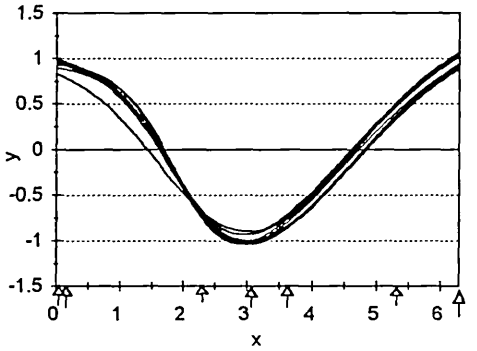
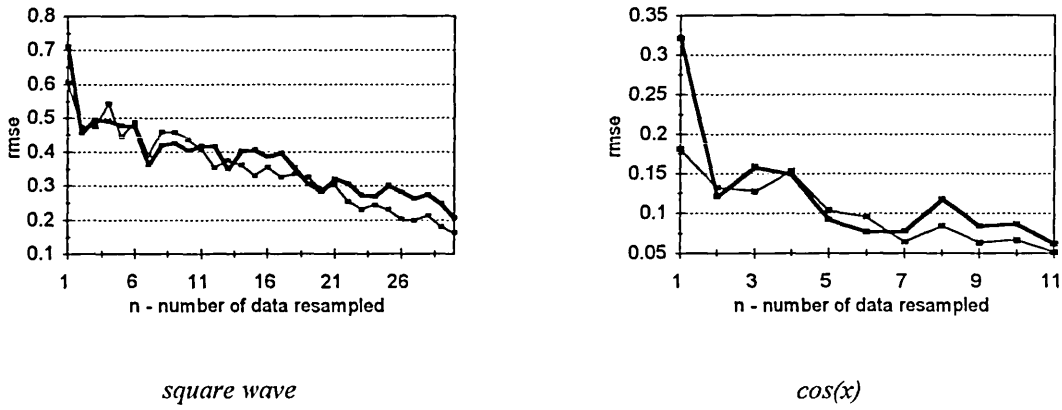


Figure 7.7.1-1(d): 10 final "agreeing" models for cos(x) after 7 resampled data points using the MIQR criterion

7.7.2 Generalisation Comparison

Figure 7.7.2-1 indicates generalisation performance (root mean square error) for cos(x) and the square wave using Maximum Variance and MIQR respectively. There is

evidence of improved generalisation using MIQR rather than Maximum Variance although the improvement is not very great.



square wave *cos(x)*

Figure 7.7.2-1 : There is evidence of improved generalisation performance using MIQR rather than using Maximum Variance although it is not very great. The thick line is the Maximum Variance curve and thin line is the MIQR curve.

7.7.3 Network Complexity and Learning Efficiency Comparison

The learning efficiency is defined as the ratio of exemplars to number of weights. Baum and Haussler [7] and Maass [69] point out that there is a close relationship between the size of a network and the number of exemplars required for good generalisation. Therefore the learning efficiency stands for the efficiency of a method for selecting exemplars on neural networks. According to Baum's theory, the lower the ratio, the more efficient the method for selecting exemplars. Table 7.7.3-1 gives the results of experiments using the MIQR method and Maximum Variance method, from which it is shown that when approaching the continuous function, the network complexity is simpler or smaller, whereas a more complicated network architecture is required when approaching the discontinuous function. Moreover, Maximum Variance consumes more resources than MIQR in order to reach the same network learning efficiency because it requires more data selection and more complicated network complexity. From here we can see again the network complexity is dependent upon the learning task, having nothing to do with the method for selecting examples. This result also confirms that it is more difficult to approach a discontinuous function than to approach a continuous function.

Table 7.6.3-1 The Ratio of Exemplars to Weights

Networks	MIQR for the square wave (hidden units h)	MV for the square wave (hidden units h)	MIQR for cos(x) (hidden units h)	MV for cos(x) (hidden units h)
1	6	7	2	2
2	7	9	2	2
3	8	7	2	2
4	7	8	2	2
5	5	6	2	2
6	7	8	2	2
7	7	7	2	2
8	6	7	2	2
9	8	6	2	2
10	7	8	2	2
Average hidden units h	6.1	7.3	2	2
Number of weights $w = 3 \times h+1$	19.3	22.9	7	7
Number of data resampled	29.2	35	7.8	8.8
Ratio of data resampled to number of weights	1.50	1.53	1.11	1.26

Ratio of data resampled to number of weights stands for the network learning efficiency. Maximum Variance consumes more resources than MIQR in order to reach the same network learning efficiency.

7.7.4 Resampling Efficiency Comparison

The Maximum Variance method is also directly derived from statistics. Theoretically, exemplar selection by the Maximum Variance method is essentially based on the assumption that values used in the computation of variance should be normally-distributed. However, in practice, the frequent presence of outliers in network outputs indicates that this is not the case. As a result, in many cases, when using Maximum Variance method, the training may be trapped in a "dead valley" and the learning procedure repeatedly locates the same data point due to the disproportionate influence of outliers on the variance. In practice, it is found that the Maximum Variance method wastes a large number of useless iterations to "discover" a new useful example, whereas the MIQR method is more efficient.

Resampling efficiency is defined as the ratio of the number of data resampled to total training iterations, standing for training procedure efficiency. Table 7.4.4-1 gives details of the amount of data selection, total training iterations and useless iteration using MIQR and Maximum Variance respectively. From this table, it is very clear that the Maximum

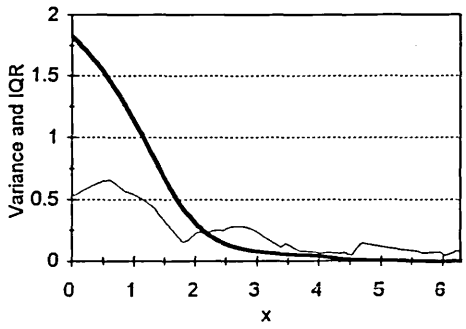
Variance method is reasonably efficient for the continuous function (cos(x)) with a ratio of number of data resampled to total iterations being 65.5% while it is quite poor for the discontinuous function (the square wave) with a ratio of number of data resampled to total iterations being only 46.2%. In contrast, the MIQR method is more efficient no matter whether approaching the continuous function or the discontinuous function with a ratio of data resampled to total iterations being 82.9% for the continuous function and 71.5% for the discontinuous function.

Table 7.4.4-1 Ratio of Number of Data Resampled to Total Training Iterations

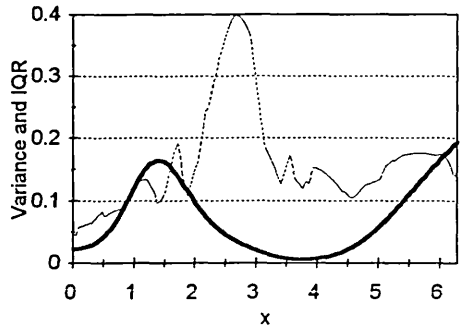
	MIQR (square wave)			MV (square wave))			MIQR (square wave)			MV (square wave)		
exp	num	ite	uls	num	ite	uls	num	ite	uls	num	ite	uls
exp1	27	38	11	30	69	39	7	8	1	11	15	4
exp2	34	48	14	38	75	37	8	10	2	8	13	5
exp3	31	44	13	37	82	45	7	7	0	8	15	7
exp4	28	40	12	36	79	43	7	10	3	10	14	4
exp5	26	34	8	34	75	41	10	12	2	7	10	3
Total	146	204	58	175	380	205	39	47	8	44	67	23
Ratio (%)	71.5			46.2%			82.9			65.5		

1. num - number of data resampled;
2. ite - total training iterations;
3. usl - useless cycles.

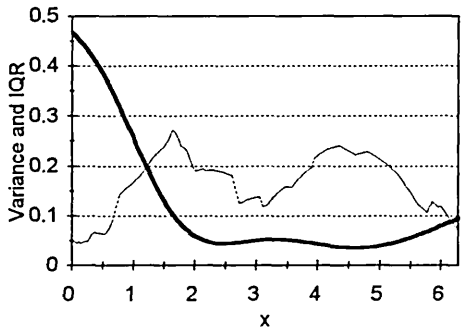
This fact can be illustrated further through details of the data selection procedure. Here, experiment 3 is given as an example. For the continuous function, the whole procedure is demonstrated. For the discontinuous function, only the earliest 10 data points are demonstrated due to too many data points being resampled and too many total training iterations for the whole procedure. From Figure 7.4.4-1 it is shown that Maximum Variance method requires 15 iterations to complete the selection of 8 data points, 7 iterations repeating previous data points and being useless iterations, whereas MIQR requires 7 iterations to complete the selection of 7 data points without repeating previous data points and useless iterations. Of course, this is a special case, in the other 4 experiments, there were 1 ~ 3 useless iterations. In the following diagrams, the thick curve indicates the "criterion line" (MIQR or Maximum Variance) by which data points are selected, and x_{MV} indicates that data points are selected using Maximum Variance method and x_{MIQR} indicates that data points are selected using the MIQR method.



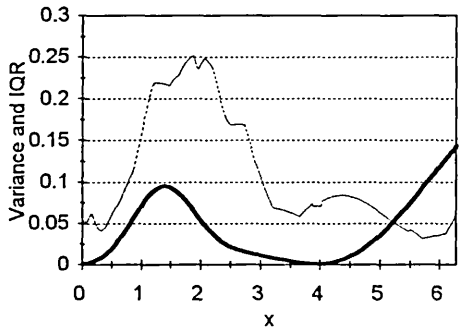
iteration = 1, x_{MV} = 0.0000
(useful, resampled)



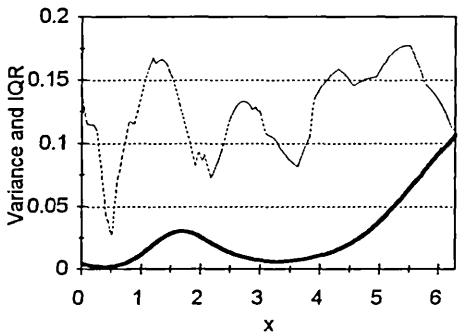
iteration = 2, x_{MV} = 6.2832
(useful, resampled)



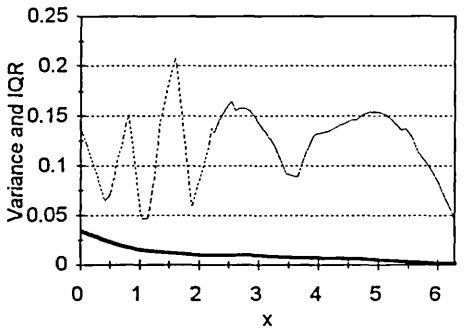
iteration = 3, x_{MV} = 0.0000
(repeated, useless)



iteration = 4, x_{MV} = 6.2832
(repeated, useless)

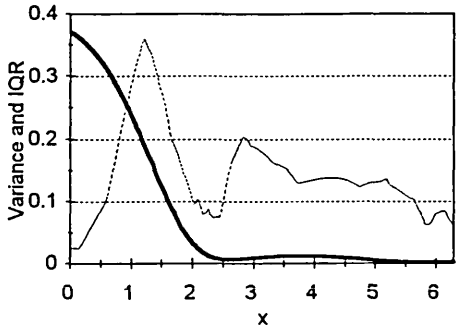


iteration = 5, x_{MV} = 6.2832
(repeated, useless)

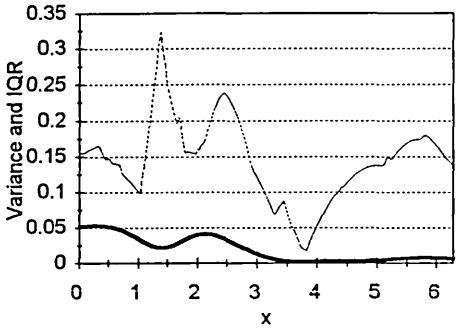


iteration = 6, x_{MV} = 0.0000
(repeated, useless)

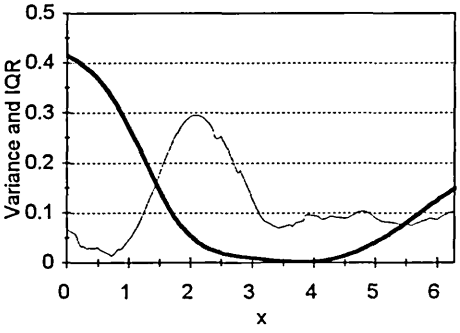
Figure 7.4.4-1(a): Exemplar selection using the Maximum Variance method for the cos(x)
(to be *continued)*



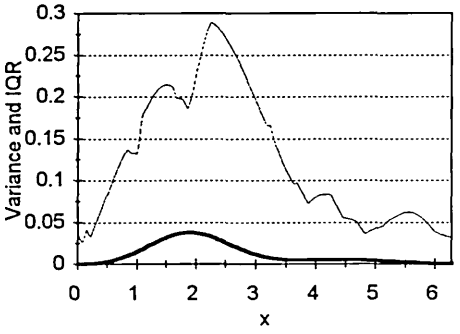
iteration = 7, x_{MV} = 0.0000
(repeated, useless)



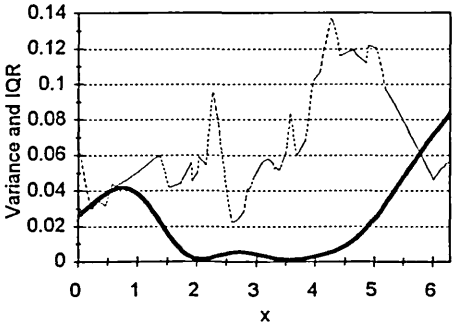
iteration = 8, x_{MV} = 0.2732
(useful, resampled)



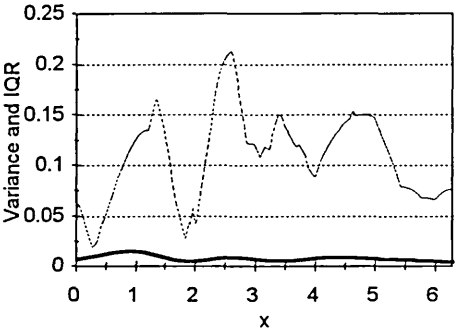
iteration = 9, x_{MV} = 0.0000
(repeated, useless)



iteration = 10, x_{MV} = 1.8913
(useful, resampled)

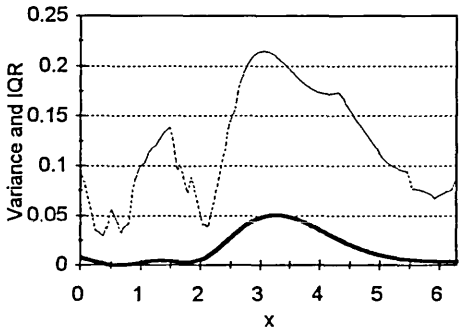


iteration = 11, x_{MV} = 6.2832
(repeated, useless)

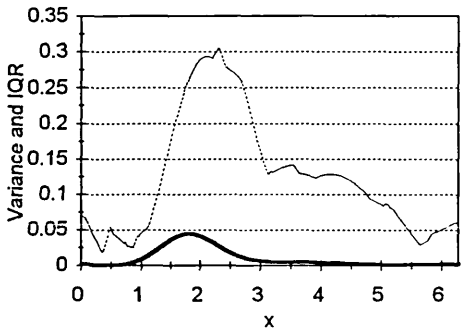


iteration = 12, x_{MV} = 0.9036
(useful, resampled)

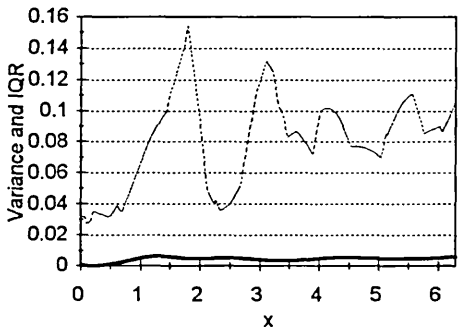
Figure 7.4.4-1(a): Exemplar selection using the Maximum Variance method for the cos(x)
(to be *continued*)



iteration = 13, x_{MV} = 3.2362
(useful, resampled)

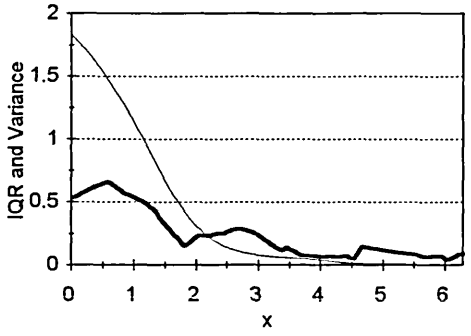


iteration = 14, x_{MV} = 1.8072
(useful, resampled)

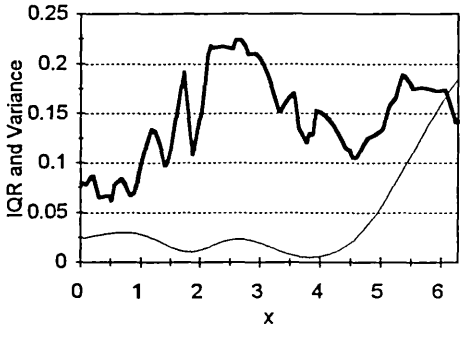


iteration = 15, x_{MV} = 1.3029 (useful, resampled)

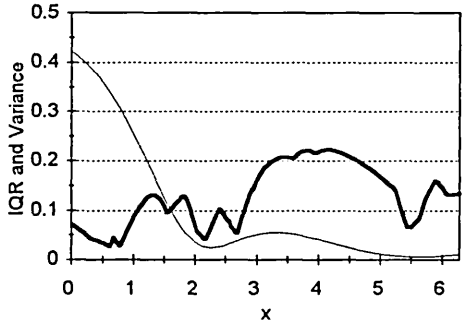
Figure 7.4.4-1 (a): Maximum Variance method requires 15 iterations to complete the selection of 8 data points for cos(x). The thick line is the variance curve and the broken line is the Inter-Quartile Range curve of outputs of the ensemble.



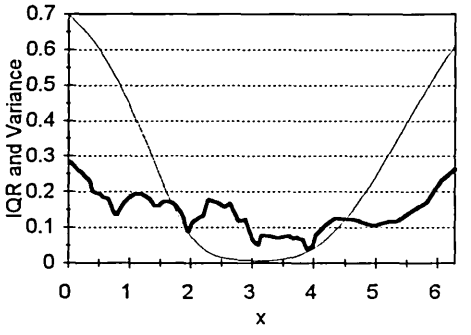
iteration =1, x_{MIQR} = 0.5884
(useful, resampled)



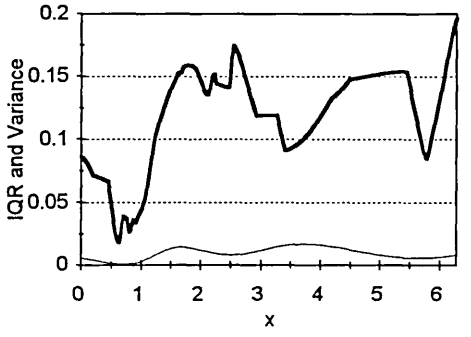
iteration =2, x_{MIQR} = 2.6268
(useful, resampled)



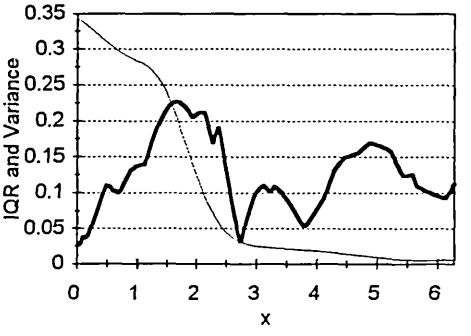
iteration =3, x_{MIQR} = 4.1398
(useful, resampled)



iteration =4, x_{MIQR} = 0.0000
(useful, resampled)

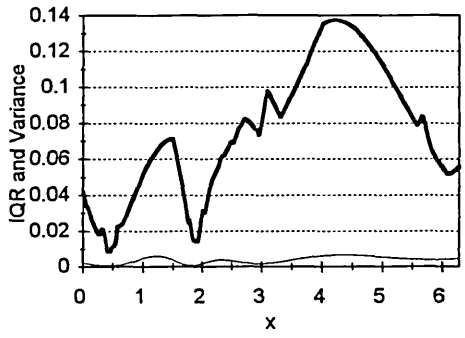


iteration =5, x_{MIQR} = 6.2832
(useful, resampled)



iteration =6, x_{MIQR} = 1.6601
(useful, resampled)

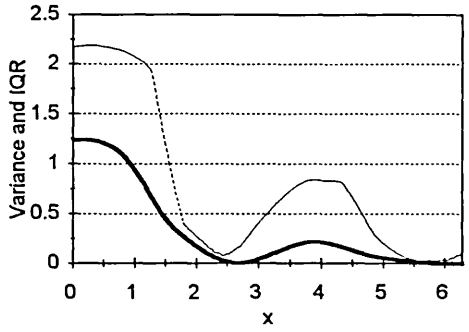
Figure 7.4.4-1(b): Exemplar selection using the MIQR method for the cos(x)
(to be continued)



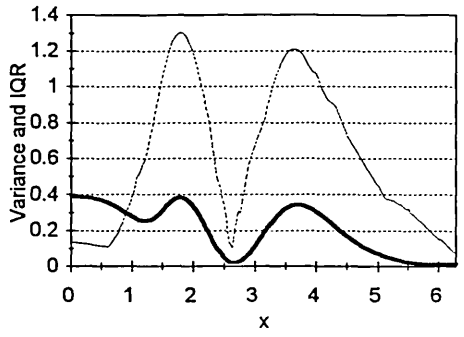
iteration =7, x_{MIQR} = 4.2.28 (useful, resampled)

Figure 7.4.4-1 (b): MIQR method requires 7 iterations to complete the selection of 7 data points for cos(x). The thick line is the Inter-Quartile Range curve and the broken line is the variance curve of outputs of the ensemble.

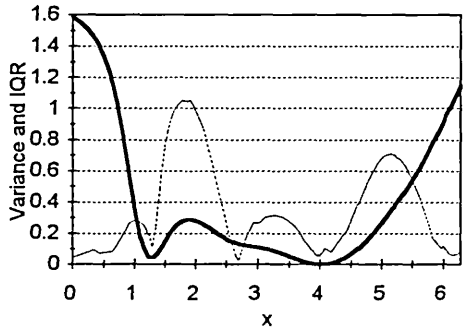
From Figure 7.7.4-2, we can see that the Maximum Variance method has spent 16 iterations to complete the selection of the first 10 data points for the square wave, with 6 iterations repeating previous data points and being useless, whereas MIQR method has spent 12 iterations to complete the selection of the first 10 data points for the square wave, with only 2 iterations repeating previous data points and being useless. It should be noted that in the earlier stages of the data selection procedure, it is easier to "discover" new information. As training progresses, it becomes more and more difficult to "discover" new information, as a result, useless iterations will increase more and more irrespective of whether the Maximum Variance method or the MIQR method is used. In general, however, totally useless iterations using the MIQR method are fewer than that using the Maximum Variance method because the MIQR method is more effective than Maximum Variance. The reason for this is that the MIQR method takes advantage of "the minority obeys the majority" philosophy, embodying the majority's desire (inter-quartile range, the mainstream of the ensemble) without considering the minority (outliers). Rather, the Maximum Variance method take into account all "outliers" of the ensemble, thus, the minority (outliers) harms the entirety's interest.



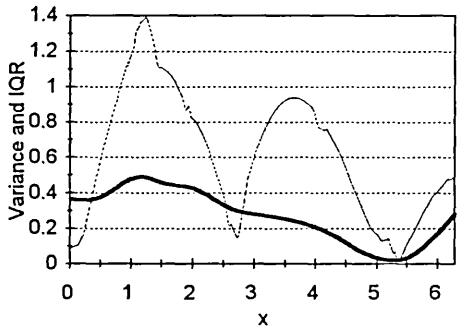
iteration = 1, x_{MV} = 0.1682
(useful, resampled)



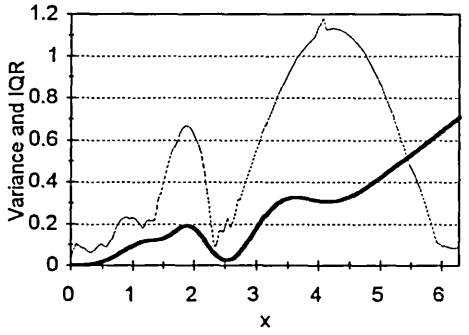
iteration = 2, x_{MV} = 0.0630
(useful, resampled)



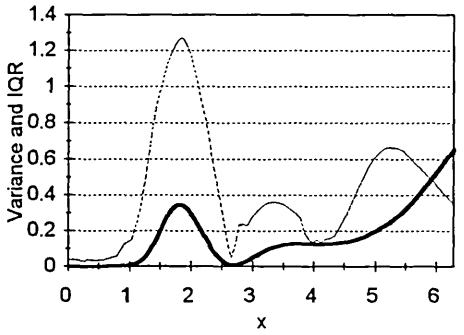
iteration = 3, x_{MV} = 0.0000
(useful, resampled)



iteration = 4, x_{MV} = 1.1558
(useful, resampled)

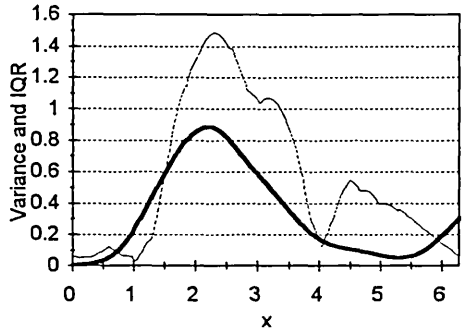


iteration = 5, x_{MV} = 6.2832
(useful, resampled)

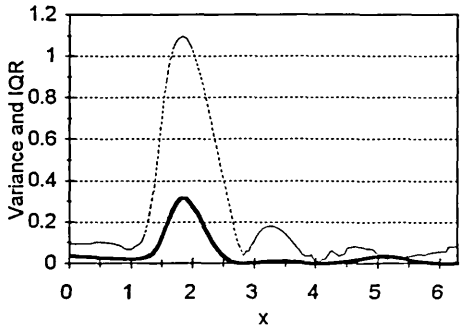


iteration = 6, x_{MV} = 6.2832
(repeated, useless)

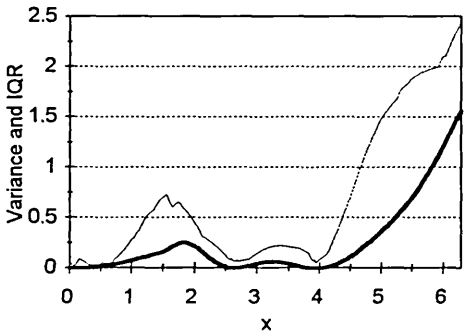
Figure 7.4.4-2(a): Exemplar selection using the Maximum Variance method for the square wave
(to be continued)



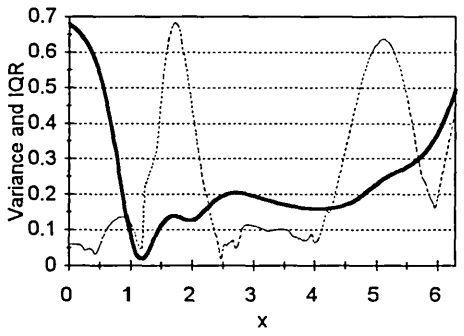
iteration = 7, x_{MV} = 2.2065
(useful, resampled)



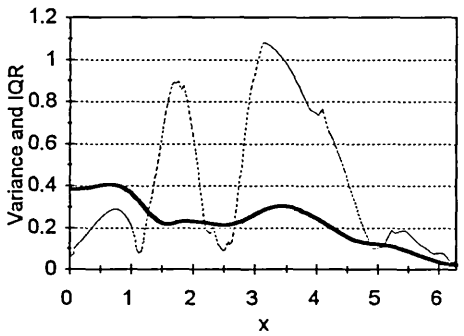
iteration = 8, x_{MV} = 1.8492
(useful, resampled)



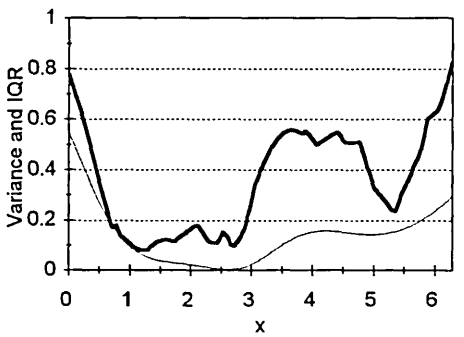
iteration = 9, x_{MV} = 6.2832
(repeated, useless)



iteration = 10, x_{MV} = 0.0000
(repeated, useless)

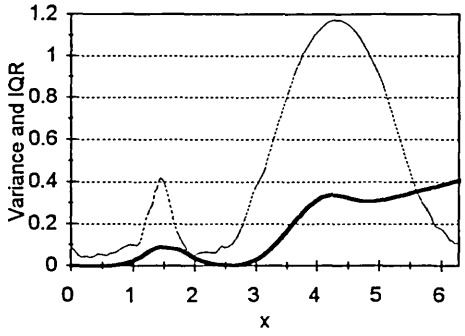


iteration =11, x_{MV} = 0.6514
(useful, resampled)

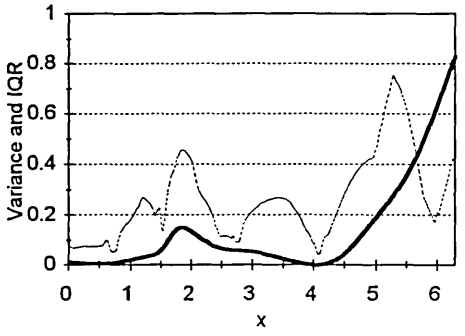


iteration =12, x_{MV} = 6.2832
(repeated, useless)

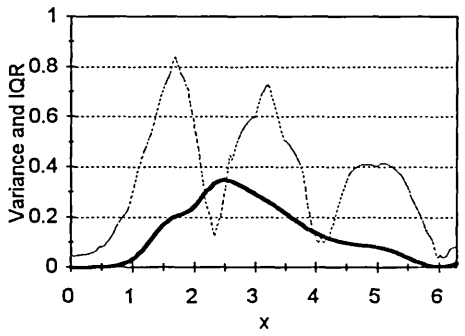
Figure 7.4.4-2(a): Exemplar selection using the Maximum Variance method for the square wave
(*to be continued*)



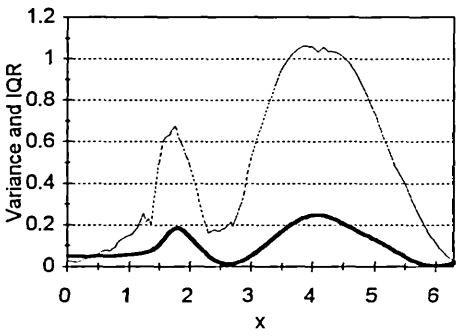
iteration =13, x_{MV} = 6.2832
(repeated, useless)



iteration =14, x_{MV} = 6.2832
(repeated, useless)

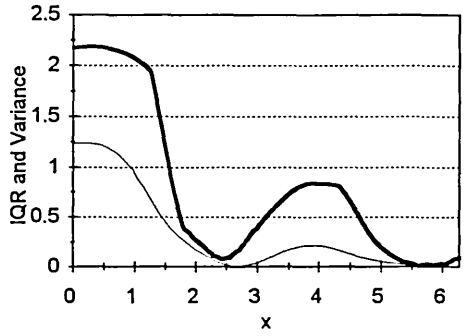


iteration =15, x_{MV} = 2.4797
(useful, resampled)

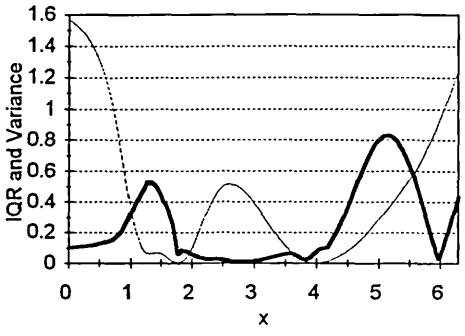


iteration =16, x_{MV} = 4.0557
(useful, resampled)

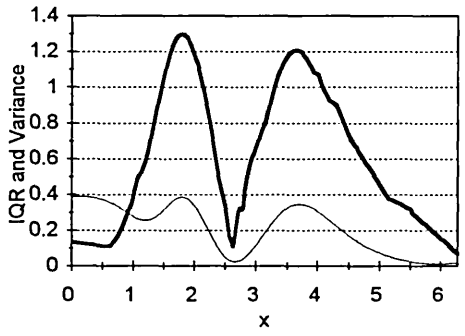
Figure 7.4.4-2 (a): Maximum Variance method has spent 16 iterations to complete the selection of the earliest 10 data points for the square wave. The thick line is the variance curve and the broken line is the Inter-Quartile Range curve of outputs of the ensemble.



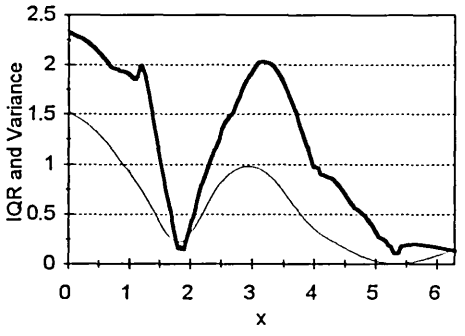
iteration =1, x_{MIQR} = 0.2732
(useful, resampled)



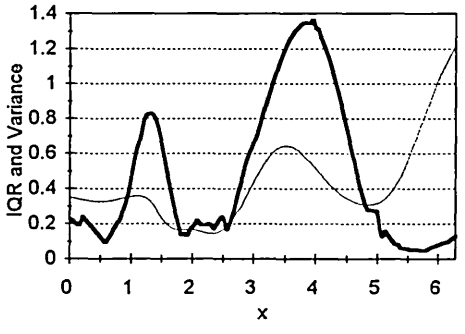
iteration =3, x_{MIQR} = 5.1274
(useful, resampled)



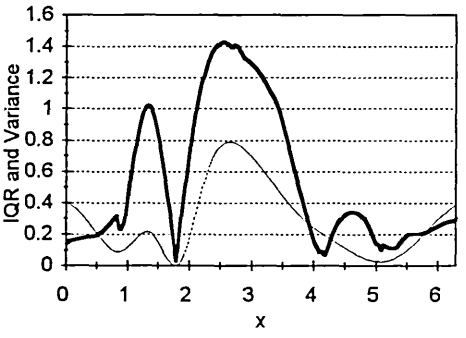
iteration =2, x_{MIQR} = 1.7862
(useful, resampled)



iteration =4, x_{MIQR} = 0.0000
(useful, resampled)

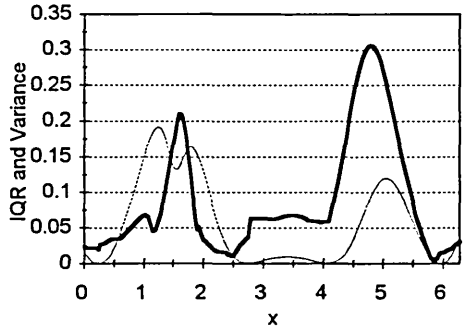


iteration =5, x_{MIQR} = 3.9506
(useful, resampled)

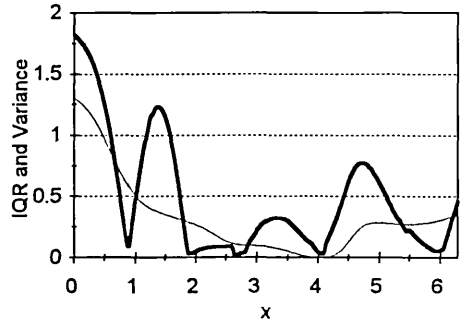


iteration =6, x_{MIQR} = 2.5427
(useful, resampled)

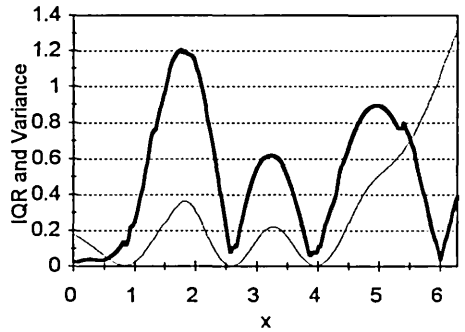
Figure 7.4.4-2(b): Exemplar selection using the MIQR method for the square wave
(to be continued)



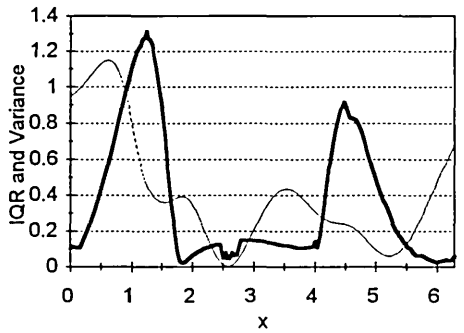
iteration =7, x_{MIQR} = 4.7702
(useful, resampled)



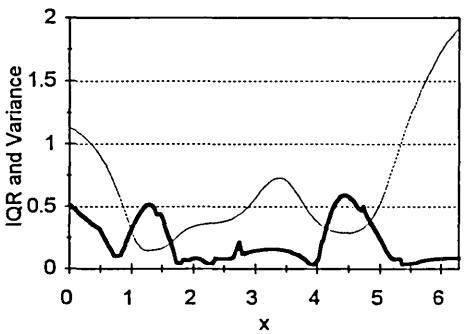
iteration =8, x_{MIQR} = 0.0000
(repeated, useless)



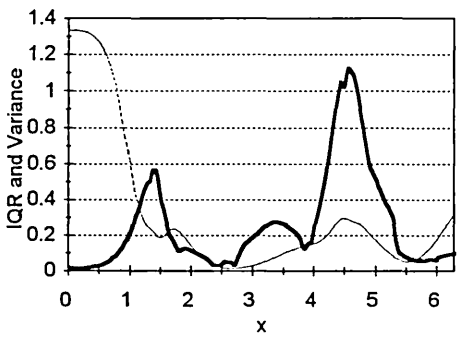
iteration =9, x_{MIQR} = 1.7442
(useful, resampled)



iteration =10, x_{MIQR} = 1.3852
(repeated, useless)



iteration =11, x_{MIQR} =4.4340
(useful,resampled)



iteration =12, x_{MIQR} =4.5600
(useful, resampled)

Figure 7.4.4-2 (b): MIQR method has spent only 12 iterations to complete the selection of the earliest 10 data points for the square wave. The thick line is the Inter-Quartile Range curve and the broken line is the variance curve of outputs of the ensemble.

7.5 Conclusion

From the above experiments the following conclusions may be drawn.

1. Training is more difficult for a discontinuous function (the *square wave*) than a continuous function (*cos(x)*). This is corroborated by three facts. Firstly, the number of data points collected for approximating to a discontinuous function is more than those for approximating to a continuous function in both active learning and passive learning. Secondly, It requires a more complicated network architecture for a discontinuous function than for a continuous function. Thirdly, it requires more training iterations for a discontinuous function than for a continuous function.

2. Active learning (MIQR) can dramatically reduce data selection. The data collected by active learning (MIQR) are fewer than those needed by passive learning no matter whether the function is continuous or discontinuous. In the active learning experiments, an average of 29.2 data points were needed to train the *square wave*; an average of 7.8 data points were needed to train *cos(x)*; On the other hand, in the passive learning experiments, an average of 65.4 data points were needed to train the same *square wave*, and an average of 19 data points were needed to train the same *cos(x)*.

3. A very interesting fact is that in the case of active learning (MIQR), the resampled data points condense near the "discontinuity". The reason for this is that to approach a continuous function, any data point is useful and therefore it is easier to quickly approximate to the function. On the other hand, to approach a discontinuous function, those data points which are near the "discontinuity" are particularly useful. For example, in our first experimental demonstration, 65 data points were resampled for passive learning whilst only 27 data points were resampled for active learning (MIQR), of which 23 data points condensed near the discontinuity. The principal aim of ***active learning*** (MIQR) is to locate such ***useful information*** without wasting time and expenditure, but maintaining the learning system's performance.

4. Most importantly, Compared with the contending method, Maximum Variance, the MIQR method is more efficient or effective in training, especially when approaching a discontinuous function (the square wave). There is also evidence of improved data selection and generalisation performance. The MIQR method takes advantage of the inter-quartile range (the mainstream of the ensemble) and ignores "outliers", reflecting "the minority obeys the majority" philosophy, rather, the Maximum Variance method take into account "outliers" of outputs of the ensemble. This suggests that an example with maximal variance may not necessarily be the most informative.

5. The network complexity is dependent upon the learning task, having nothing to do with the method for selecting examples (no matter whether active learning or passive learning). The experimental results show that for the continuous function (cos(x)), the network architecture of 2 hidden units is sufficient to train it well, whereas for the discontinuous function (the square wave), a network architecture of 6 ~ 7 hidden units is required.

From the above, we can see that it is "the minority obeys the majority" philosophy that plays a key role in MIQR active learning. Consequently, a question naturally arises of the number of networks of which the ensemble consists that will achieve the best result from the "the minority obeys the majority" philosophy, namely, the fairness problem. In Chapter 6 and this Chapter, the ensemble is composed of 10 networks. According to common sense, the more the voter population, the more fairness in the election. But in the neural network ensemble, this is restricted by linear independence of networks. In the next chapter this problem will be discussed in detail.

Chapter 8

Sensitivity Analysis of MIQR Based on Ensemble Networks

8.1 Summary

We have in previous chapters proposed a new effective criterion for exemplar selection, MIQR, which is directly derived from statistics. As discussed in detail in Chapter 7, MIQR is more efficient or effective than the contending Maximum Variance method. We also provided a complete algorithm based on a set of networks (ensemble). One issue should be discussed further. In previous chapters, such an ensemble or committee has been composed of 10 networks. This chapter provides further exploration of how ensemble methods improve generalisation performance. In Chapter 4 and 5, we presented ensemble methods, i.e. how to create an ensemble and how to combine members (networks) of the ensemble. Here, the results of a sensitivity analysis are presented. This sensitivity analysis is concerned with following questions. Firstly, how many networks in an ensemble give the best generalisation performance? Secondly, What changes in data selection result? Finally, how does it influence the network complexity and total training iterations?

Analytical results indicate that as the ensemble is growing, generalisation performance is improved. However, this improvement will reach a limit. In general, 20 networks is maximal (see Chapter 4).

8.2 The Experimental Comparisons

We performed a number of experiments, with the number of ensemble members being 5, 10, 15 and 20 networks respectively, based on the same initial conditions (so that we have the same data distributions). For each comparison we ran 5 sets of experiments and then averaged these results.

Now that number of members in the ensemble is different, the computation of Inter-Quartile Range needs be updated, being indicated in Table 8.2-1, here Y^* is the rearrangement of Y in ascending order as discussed in Chapter 6.

Table 8.2-1 Computation of Inter-Quartile Range

Number in ensemble	Inter-Quartile Range (x_i)	Outputs of ensemble networks
5	$Y_4^*(x_i) - Y_2^*(x_i)$	Y_1, Y_2, Y_3, Y_4, Y_5;
10	$Y_8^*(x_i) - Y_3^*(x_i)$	$Y_1, Y_2, Y_3, ..., Y_9, Y_{10}$;
15	$Y_{12}^*(x_i) - Y_4^*(x_i)$	$Y_1, Y_2, Y_3, ..., Y_{14}, Y_{15}$;
20	$Y_{16}^*(x_i) - Y_5^*(x_i)$	$Y_1, Y_2, Y_3, ..., Y_{19}, Y_{20}$.

8.2.1 Evidence of Reduced Data Selection

Table 8.2.1-1 (a) and (b) are the results of data selection for the continuous function and the discontinuous function respectively, from which it is shown that as the number of members of the ensemble grows, the amount of data resampled decreases. According to the MIQR method, it is easily understood that the bigger the ensemble becomes, the more reasonable or justified the algorithm becomes, because it more faithfully reflects "the minority obeys the majority" philosophical idea. As a result, data resampled is more informative. But data selection cannot be arbitrarily improved as the number of members of the ensemble grows. It is restricted by the linear independence of networks of the ensemble.

Table 8.2.1-1(a): Data Selection for the square wave

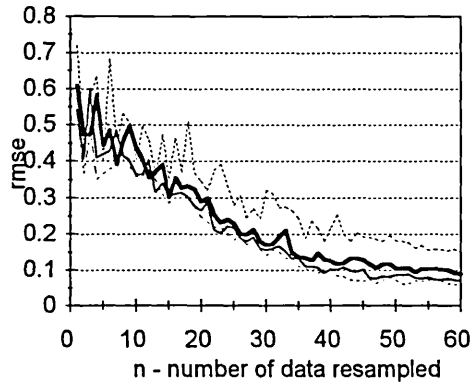
Ensemble / Experiment	Ensemble = 5 (Data Selection)	Ensemble = 10 (Data Selection)	Ensemble =15 (Data Selection)	Ensemble = 20 (Data Selection)
Exp1	33	27	26	24
Exp2	41	34	29	27
Exp3	37	32	24	29
Exp4	31	28	26	21
Exp5	34	26	27	25
Total	176	147	132	126
Average	35.2	29.4	26.4	25.2

Table 8.2.1-1(b): Data Selection for cos(x)

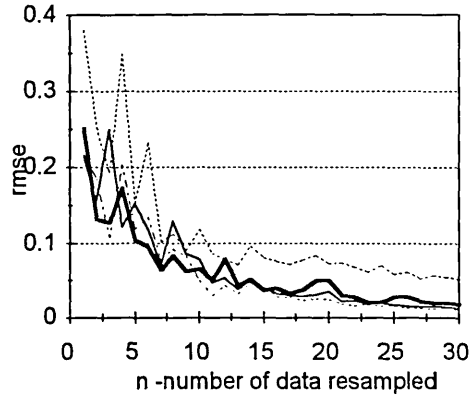
Ensemble / Experiment	Ensemble = 5 (Data Selection)	Ensemble = 10 (Data Selection)	Ensemble =15 (Data Selection)	Ensemble = 20 (Data Selection)
Exp1	9	7	7	7
Exp2	7	8	9	8
Exp3	11	7	6	7
Exp4	14	7	7	6
Exp5	8	10	8	8
Total	48	39	37	36
Average	9.6	7.8	7.4	7.2

8.2.2 Evidence of Improved Generalisation

As we have discussed in Chapters 4 and 5, ensemble methods can improve generalisation performance. Figure 8.2.2-1 indicates that generalisation can be improved further as the ensemble is growing. The general trend is downwards and smooth although it oscillates in the beginning of the training. But like data selection, this is restricted by the linear independence of networks of the ensemble. Our experimental result is in accordance with previous results (see Chapter 4). It is suggested from this experimental result that an ensemble consisting of 15 networks is sufficient to obtain a good generalisation performance.



square wave



cos(x)

Figure 8.2.2-1: Generalisation is measured by root mean squared error versus number of data resampled. The broken line (the uppermost line) is the ensemble consisting of 5 networks; the thick line (the second uppermost line) is the ensemble consisting of 10 networks; the thin line (the third line) is the ensemble consisting of 15 networks ; and the dot and broken line (the lowest line) is the ensemble consisting of 20 networks.

8.2.3 Evidence of Improved Network Learning Efficiency

It is shown from Table 8.2.3-1(a) and (b) that growing the ensemble also can improve network learning efficiency because growing the ensemble results in more informative data resampling, but it is restricted for the same reason as generalisation. We can see from here, network learning efficiency is considerably dependent upon the function to be mapped. In our experiments the limit of the network learning efficiency is near to 1.40 for the square wave (the discontinuous function), and 1.00 for cos(x) (the continuous function). It would probably be vain even to attempt further increasing the number of the ensemble. From this result, we can see, a network architecture with 2 hidden units is sufficient to approach cos(x) and on the other hand a network architecture with an average of 6 hidden units would be required to approach the square wave. In this sense, network complexity is dependent upon the function to be mapped, having nothing to do with number of members of the ensemble.

Table 8.2.3-1(a) The Ratio of Exemplars to Weights for the Square Wave

Ensemble / Networks	Ensemble = 5 (hidden units)	Ensemble = 10 (hidden units)	Ensemble = 15 (hidden units)	Ensemble = 20 (hidden units)
1	6	6	7	6
2	7	7	6	6
3	8	6	7	5
4	6	8	4	7
5	5	5	5	6
6		7	6	4
7		7	7	5
8		6	6	7
9		8	5	6
10		7	8	5
11			5	7
12			6	6
13			6	5
14			5	7
15			7	6
16				5
17				6
18				6
19				5
20				7
Average hidden units h	6.4	6.1	5.97	5.7
Number of weights $w = 3 \times h+1$	20.2	19.3	18.9	18.1
Number of data resampled	35.2	29.2	26.4	25.2
Ratio of data resampled to number of weights	1.74	1.51	1.40	1.39

Ratio of data resampled to number of weights stands for the network learning efficiency.

Table 8.2.3-1(b) The Ratio of Exemplars to Weights for cos(x)

Ensemble / Networks	Ensemble = 5 (hidden units)	Ensemble = 10 (hidden units)	Ensemble = 15 (hidden units)	Ensemble = 20 (hidden units)
1	2	2	2	2
2	2	2	2	2
3	2	2	2	2
4	2	2	2	2
5	2	2	2	2
6		2	2	2
7		2	2	2
8		2	2	2
9		2	2	2
10			2	2
11			2	2
12			2	2
13			2	2
14			2	2
15			2	2
16				2
17				2
18				2
19				2
20				2
Average hidden units h	2	2	2	2
Number of weights $w = 3 \times h+1$	7	7	7	7
Number of data resampled	9.6	7.8	7.4	7.2
Ratio of data resampled to number of weights	1.37	1.11	1.06	1.03

Ratio of data resampled to number of weights stands for the network learning efficiency.

8.2.4 Evidence of Improved Resampling Efficiency

Resampling efficiency is defined as the ratio of the number of data points resampled to total training iterations as discussed in Chapter 7. It is indicated from Table 8.2.4-1, growing the ensemble also can improve resampling efficiency since it results in more informative data resampling and reduces the probability of repeated data selection, and

therefore reduces useless training iterations. But as discussed above, this improvement is also restricted by the linear independence of networks of the ensemble.

Table 8.2.4-1: Resampling Efficient for the Square Wave

	Ensemble = 5			Ensemble = 10			Ensemble = 15			Ensemble = 20		
Exp	num	ite	uls	num	ite	uls	num	ite	uls	num	ite	uls
Exp1	33	55	22	27	38	11	26	33	7	24	32	8
Exp2	41	59	18	34	48	14	29	42	13	27	37	10
Exp3	37	60	23	31	44	13	24	35	11	29	40	11
Exp4	31	45	14	28	40	12	26	33	7	21	27	9
Exp5	34	51	17	26	34	8	27	36	9	25	32	7
Total	176	270	94	147	204	58	132	179	47	126	171	45
Ratio (%)	65.2 %			71.5 %			73.2 %			73.7 %		

1. num - number of data resampled;
2. ite - total training iterations;
3. usl - useless iterations;
4. Resampling efficiency is defined as the ratio of number of data resampled to total training iterations, standing for training procedure efficiency.

Table 8.2.4-1(b) Resampling Efficiency for cos(x)

	Ensemble = 5			Ensemble = 10			Ensemble = 15			Ensemble = 20		
Exp	num	ite	uls	num	ite	uls	num	ite	uls	num	ite	uls
Exp1	9	12	3	7	8	1	7	7	0	7	8	1
Exp2	7	8	1	8	10	2	9	12	3	8	10	2
Exp3	11	16	5	7	7	0	6	7	1	7	9	2
Exp4	14	17	3	7	10	3	7	7	0	6	7	1
Exp5	8	10	2	10	12	2	8	10	2	8	8	0
Total	48	62	14	39	47	8	37	43	6	36	42	6
Ratio (%)	77.4 %			82.9 %			84.1 %			85.6 %		

1. num - number of data resampled;
2. ite - total training iterations;
3. usl - useless iterations;
4. Resampling efficiency is defined as the ratio of number of data resampled to total training iterations, standing for training procedure efficiency.

8.3 Conclusion

From the above discussion, it is clearly suggested that growing the ensemble of neural networks can improve data selection, generalisation performance, network learning

efficiency and resampling efficiency. The reason for these is that growing the ensemble results in more informative data resampling because, according to the MIQR method's philosophy ,i.e., "the minority obeys the majority", growing the ensemble, in some sense, equivalently enhances the fairness of the method. This is similar to a presidential election or parliamentary decision by vote. The more the voters, the more fair the election or decision. MIQR method's principle and essence is to implement "***fairness***", reflecting the ***majority***'s desire by ignoring the ***minority*** (outliers). This is also the reason why the MIQR method is more effective than other contending methods.

However, the improvement is restricted by linear independence of the networks of the ensemble. Our experimental results are consistent with the previous result, i.e., in most cases, the number of networks used in an ensemble is no more than 20 networks (see Chapter 4). It is indicated from our experiments that an ensemble consisting of 15 networks is sufficient to effectively approach any functions - cos(x) and the square wave are a typical continuous function and a typical discontinuous function, respectively.

The above experiments also show that network complexity is dependent upon the function to be mapped, having nothing to do with size of the ensemble. Consequently, growing the ensemble cannot improve or save network complexity.

Of course, growing the ensemble must result in increasing processing time. Cohn assumes in his robotic manipulator experiment that if computation is cheap and data selection is expensive, active exploration is efficient and effective [22]. By this token, the MIQR method is feasible in applications such as oil field exploration, in which it is very important to accurately "discover" the location of a oil well. There has been no evidence indicating that processing time increases linearly with the number of members in the ensemble, which will be investigated further in the future.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

We set out to obtain an effective method for selecting exemplars for training neural networks by justifying an objective sampling criterion appropriate for use with the criterion used for learning. Although many interesting questions remain to be investigated, we have accomplished this work to a large degree.

The MIQR method for selecting exemplars is satisfying and encouraging, as it outperforms the contending methods, Maximum Variance and Maximum Error. We do not compare this method with Maximum Error method due to the fact that the Maximum Error method (for example, Plutowski's ΔISB) is only used in a single network, hence it would be hard to conduct this comparison. However, it is intuitively clear that the MIQR method has a wider range of applications than Maximum Error, since Maximum Error methods not only require the computation of a large Hessian matrix (computationally expensive), but also require the assumption that the functions to be mapped are reasonably smooth. Furthermore, as Plutowski admits, his ΔISB method has no clear stopping criterion because the Maximum Error method demands deterministic or labelled data. In contrast, MIQR takes advantage of unlabelled data, thus MIQR not only has a clear stopping criterion (through computing maximum variance of outputs of the ensemble), but is also easily extended to a noisy data environment.

MIQR is particularly appealing when approaching a discontinuous function (for example, the square wave, a typical discontinuous function), for which previous work is weak no matter whether the Maximum Variance or the Maximum Error is used. This method obviously has several important benefits in more general purpose applications, being not only computationally simpler and cheaper, but also it results in more informative data being resampled, and improved generalisation performance as well. Based on ensemble methods, MIQR makes full use of "the minority obeys the majority" philosophy. Although the Maximum Variance method can also make use of the ensemble method, it takes into account all outputs including outliers which are frequently produced, in particular, when approaching a very sharp discontinuous function, for example, the square wave. Because of this, although outliers are a minority, they may influence and reduce ensemble efficiency. As a result, data resampled corresponding to the maximum variance does not mean the most informative information. In contrast, the MIQR method's principle and essence is to embody "fairness", reflecting the majority's desire without considering minority (outliers). The general mainstream (within the inter-quartile range) is not influenced by outliers, therefore, ensemble efficiency is assured. Furthermore, previous work with Maximum Variance does not present a complete algorithm, the training threshold and network architecture being fixed for the whole training procedure. This results in apparent deficiencies. First, a stringent training threshold would consume unnecessary resources. On the other hand, a loose training threshold is not able to discover new information. Second, a big network architecture not only consumes too much system resource, but also may easily lead to the "oversaturated" state. On the other hand, a simple network architecture is insufficient to fit the training data set. We present a complete algorithm, which can automatically regulate the network architecture and the training threshold as necessary.

Speaking in detail, this dissertation has fulfilled following tasks: first, we have developed a new active learning method i.e. Maximum Inter-Quartile Range (MIQR). The MIQR is directly derived from nonparametric statistics, being computationally simple and cheap. In addition, data selection is not influenced by the "outliers", instead it is dependent upon the

general trend of the ensemble i.e. the mainstream of the majority (inter-quartile range). In this sense, the MIQR active learning method is based on the "fairness" - "the minority obeys the majority" philosophical idea. As a result, data selection by the MIQR method not only has become more fair, but also become more informative than by the existing active learning methods. Furthermore, it is the "fairness" philosophy in which the MIQR method lies, which make the MIQR active learning more effective and efficient, even when it approaches a typical discontinuous function. From the standpoint of methodology, the MIQR is a theoretical breakthrough in active learning in the neural network community.

Second, in the MIQR active learning algorithm, unlabelled data is validated, therefore this method not only has a clear stopping criterion, but also can be directly extended to a noisy data environments. It should be also noted that we ingeniously combine advantages of the maximum inter-quartile range and the maximum variance, namely, we use the maximum inter-quartile range as the data selection criterion whereas we use the maximum variance as the stopping criterion (see the Section 6.5 Discussion of Chapter 6).

Third, we have developed a complete algorithm, in which the neural network complexity and the training threshold for individual networks are automatically regulated as necessary. Therefore we not only save considerable resources, but also avoid neural network training's "oversaturated" or "undersaturated" problems, with which previous work is not able to deal [59][88].

Fourth, previous work is only concerned with data selection and generalisation. Our work not only has discussed data selection and generalisation in detail, but also has explored inside the network architectures and training procedures through examining the network learning efficiency and the resampling efficiency. Therefore this work has become more profound in active learning methodology.

Finally, previous work has not presented a sensitivity analysis of active learning methods based on ensemble networks. We present a detailed analysis of the MIQR based on

ensemble networks. The experimental results are not only significant and encouraging, but also are interesting, and are in accordance with theoretical results as well.

In a word, MIQR's contribution and importance lies in that it not only plays a role in active learning, but also it outperforms current contending methods (Maximum Variance and Maximum Error), and may be the most effective active learning technique. It combines the advantages of both the ensemble methods and the active learning methodology. Since MIQR is directly derived from nonparametric statistics, its theoretical justification is beyond question. Although MIQR has been used on clean data selection, it can be directly extended to noisy data selection as it stands, since MIQR uses unlabelled data. In particular, the thinking behind MIQR is significant for other scientific research, educational methodology and societal behaviours.

9.2 Future Work

Due to the richness of this research, many questions are left for future work. Firstly, we should investigate the MIQR method in a different bandwidth of the input space. In above experiments, we fixed the environmental distribution of 300 data points. Future work should investigate experimental results on the environmental distribution of different numbers of data points, such as 100, 200, 400 and 500 data points.

Secondly, in the experiments with the MIQR technique, it has been found that there are many local maximal points in the Inter-Quartile Range as indicated in Figure 9.2-1. In fact, the MIQR criterion is optimised by the backpropagation algorithm based on minimising the least square error of the target function and averaging the output of the ensemble networks, resulting in several local maxima (or local minima). We conjecture that the second or third maximal points may be the points which will have the largest inter-quartile range in the next iterations of data selection. If this conjecture is justified, MIQR can simultaneously select several data points at a time, rather than just selecting one data

point. This will reduce the task of searching for new data points and considerably speed up training.

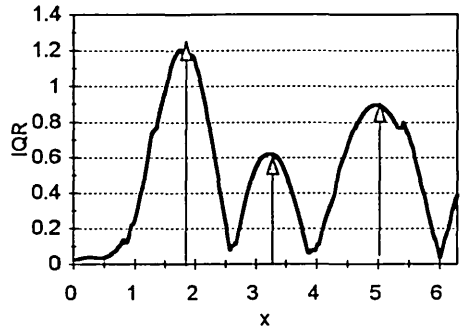


Figure 9.2-1: There are several local maximal points in IQR.

Thirdly, the definition and computation of the formula of Inter-Quartile Range is directly taken from the nonparametric statistics. It may be modified to obtain variants of the method for practical use on particular applications. Recall the definition of the outliers and inter-quartile range: the ratio of number of the outliers and inter-quartile range will influence resulting data selection. Because MIQR's philosophy is "the minority obeys the majority", thus, the limits defining the minority (outliers) and the majority (inter-quartile range) undoubtedly play an important role in MIQR active learning. When obtaining the most concise set of exemplars is the highest priority, we expect that delimiting outliers directly will be worthy of investigation.

Fourthly, according to the MIQR method's philosophy ,i.e., "the minority obeys the majority", growing the ensemble, equivalently enhances the fairness of the method. This means that more the number of networks in an ensemble, the more fair the MIQR method for selecting exemplars. However, increasing size of the ensemble, not only the improvement in terms of data selection, generalisation performance, learning efficiency and resampling efficiency is restricted by linear independence of the networks of the ensemble as discussed in Chapter 8, but also, more importantly, increasing size of the ensemble will increase the consumption of more resources. It would be significant if we

could find out a "balance point" at which the MIQR method attains in both the maximal "fairness" and minimal resource. So far, the size of the ensemble has been fixed in our experiments. Further work should study how to create a criterion by which the size of the ensemble could be flexible so that it could be automatically regulated as necessary, like the network complexity and the training threshold.

Finally, we have summarised our initial experience with the technique on difficult learning tasks. Many other experiments are possible and we intend to apply the technique to learning tasks requiring networks with multiple inputs and outputs.

REFERENCES and BIBLIOGRAPHY

1. Ahmand, S. and Omohundro, S. (1990). "A network for extracting the locations of point clusters using selective attention." Technical Report TR 90-011, International Computer Science Institute, University of California, Berkeley.

2. Al-Ghoneim, K. and Vijaya Kumar, B.V.K. (1995) " Learning ranks with neural networks." In *Applications and Science of Artificial Neural Networks, Processings of the SPIE*, vol. 2492, pp.446-464.

3. Ash, A. and Hedayat, A. (1978) "An introduction to design optimality with an overview of the literature." *Communications in Statistics, Part A - Theory and Methods*, 7, 1295-1325.

4. Atkinson, A.C. and Donev, A.N. (1992). **Optimal Experimental Designs**. Oxford Science publications.

5. Barnett, J.A.(1981) "Computational methods for a mathematical theory of evidence." *In processings of IJCAI*, pp.868-875.

6. Barnett, V. and Lewis, T. (1993) "Outliers in Statistical Data." (2nd edition). John Wiley & Sons, New York.

7. Baum, E.B. and David Haussler. (1989). "What size network gives valid generalisation." In Touretzky, D. (ed.), *Advances in Neural Information Processing Systems 1*, San Mateo, CA: Morgan Kaufmann Publishers.

8. Baum, E.B. (1991). "Neural net algorithms that learn in polynomial time from examples and queries." *IEEE Transactions on Neural Networks*. **2**, 1, pp5-19.

9. Beran, R. (1988). "Prepivoting Test Statistics: A Bootstrap View of Asymptotic Refinements." *Journal of the American Statistical Association*, vol. 83, pp.687-697.

10. Bishop, C. (1995). **Learning and Generalisation**, Chapter 9 in *Neural Networks for Pattern Recognition*, Oxford University Press.

11. Box, G. and Tiao, G.C.(1968) "A Bayesian approach to some outlier problems." *Biometrika*, 55, 119-129.

12. Box, G. and Draper, N. (1987). **Empirical Model-Building and Response Surfaces**. Wiley, New York.

13. Breiman, L. (1993) "Stacked regression." *Technical Report 367*, University of California, Berkeley

14. Breiman, L. (1994). **Bagging predictor**. *Tech. Report, TR-421,* Dept. of Statistics, University of California, Berkeley.

15. Breiman, L. (1996a) "Bias, Variance and Arcing Classifiers." *Technical Report 460,* University of California, Berkeley

16. Breiman, L. (1996b) "Bagging predictors using bootstrapping techniques." *Machine Learning*, **24**, 123-140.

17. Bruce, E. Rosen (1996). "Ensemble Learning Using Decorrelated Neural Networks." *Connection Science, Vol.8, Nos 3 & 4* , 373-383.

18. Cachin, C. (1994). "Pedagogical pattern selection strategies." *Neural Networks*, 7(1): 175-181.

19. Clemen, R. (1989) "Combining forecasts: a review and annotated bibliography." *International Journal of Forecasting*, **5**, 559-583.

20. Cohn, D. (1990) "A local approach to optimal queries." *Connectionist Models Summer School (CMSS-90).* Proc. of the 1990 Summer School in San Diego. David S. Touretzky, et al., ed. Morgan Kaufmann, San Mateo, California.

21. Cohn, D., Atlas, L., and Ladner, R. (1990) "Training connectionist networks with queries and selective sampling." *Advances in Neural Information Processing Systems 2,* Proc. of The Neural Information Processing Systems Conference. Morgan Kaufmann, San Mateo, California.

22. Cohn, D. (1994). "Neural network exploration using optimal experiment design." *Advances in Neural Information Processing Systems 6,* MIT press, Cambridge MA.

23. Cohn, D., Ghahramani, Z. and Jordan, M. I. (1995). "Active learning with statistical models." *Advances in Neural Information Processing Systems 7,* Cambridge, MA, MIP Press.

24. Conover, W.J. (1980) "Practical Nonparametric Statistics." *Texas Tech University.* John Wiley & Sons, New York.

25. Cooray-Wijesinha, M. and Khuri, A.I. (1987). "The sequential generation of multiresponse D-optimal designs when the variance-covariance matrix is not known." *Comm. Statistics - Simulation,* **16**(1), 239-259.

26. Cover, T. M. (1965). "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern recognition." *IEEE Transactions on Electronic Computers,* EC-14, 326-334.

27. Daniel, W. W. (1978) " Applied Nonparametric Statistics." *Georgia State University.* Houghton Mifflin Company, Boston, USA.

28. Darrow, S. C., Opitz, D. W. and Shavlik, J. W. (1996) "Actively Searching for an Effective Neural Network Ensemble." *Connection Science, Vol.8, Nos 3 & 4* , 337-353

29. Eberhart, R. C. (1992). "The Role of Genetic Algorithms in Neural Network Query-Based Learning and Explanation Facilities." *IEEE Computer Society Press*, CA, USA.

30. Efron, B. (1982). "The Jackknife, the Bootstrap, and Other Resampling Plans." *Society for Industrial and Applied Mathematics,* Philadelphia, PA.[110]

31. Efron, B. and Tibshirani, R. (1993). **An Introduction to the Bootstrap**. Chapman and Hall, New York.

32. EI-Gamal, M.A. (1991). "The role of priors in active Bayesian learning in the sequential statistical decision framework." *In Maximum Entropy and Bayesian Methods*, W.T. Grandy, Jr. and L.H. Schick, eds., pp.33-38. Kluwer, Dordrecht.

33. Faraway, J. (1990) "Sequential design for the nonparametric regression of curves and surfaces." *Technical Report #177*, Department of Statistics, The University of Michigan.

34. Fedorov, V.V. (1972). "Theory of optimal experiments." Academic Press, New York.

35. Freund, Y., Seung, H.S., Shamire, E. and Tishby, N. (1993) "Information, prediction, and query by committee." In *Advances in Neural Information processing Systems, Volume* 5, San Mateo, California.

36. Freud, Y., Seung, H. S., Shamire, E. and Tishby, N. (1995). "Selective sampling using the query by committee algorithm." *Tech. Report*, Centre for Neural Computation, Hebrew University, Jerusalem.

37. Freund, Y. and Schapire, R. (1996). "Experiments with a new bootstrapping algorithm." In *Machine Learning: Processings of the Thirteenth International Conference.*

38. Gasarch, W.I. and Pleszkoch, M.B. (1989) " Learning via queries to an oracle." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

39. Geman, G. and Doursat, R. (1992) "Neural networks and the bias/variance dilemma." *Neural Computation,* **4**(1): 1-58.

41. Granger, C.W.J. (1989) "Combining forecasts-twenty years later." *Journal of Forecasting*, **8**, 167-173.

42. Green, P.J. and Silverman, B.W. (1994) " Nonparametric Regression and Generalized Linear Models." *School of Mathematics, University of Bristol.* Chapman and Hall.

43. Hansen, L.K. and Salamon, P. (1990). "Neural network ensemble." *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**, 993-1000.

44. Hashem, S. and Schmeiser, B. (1993) "Approximating a function and its derivatives using MSE-optimal linear combinations of trained feedforward neural networks." In *processings of the World Congress on Neural Networks, Vol. 1.* New Jersey: Lawrence Erlbaum Associates, pp.617-620.

45. Hashem, S. (1996) "Effects of Collinearity on Combining Neural Networks." *Connection Science, Vol.8, Nos 3 & 4* , 315-336

46. Haussler, David. (1990). "Decision theoretic generalisations of the PAC model for neural net and other learning applications." *Technical report UCSC-CRL-91-02*, Baskin Center for Computer Engineering and Information Sciences, University of California, Santa Cruz, California.

47. Hassoun, M.H. (1995) " Fundamentals of Artificial Neural Networks." *The MIT Press*, Cambridge, MA.

48. Haussler, D., Kearns, M.J. and Schapire, R.E. (1991). "Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension." *Proc. Neural Networks for Computing*, Snowbird, Utah.

49. Hellerstein, L. and Karpinski, M. (1989). "Learning read-once formulas using membership queries." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

50. Huber, P. J. (1977) . **Robust Statistical Procedure**. John Wiley and Sons, New York.

51. Hwang, J., Choi, J. Oh and Marks, R. (1990) "Query learning based on boundary search and gradient computation of trained multilayer perceptrons." *In Proceedings IJCNN 1990, San Diego, CA, The International Joint Conference on Neural Networks.* IEEE Press.

52. Hwang, J., Choi, J. Oh and Marks, R. (1990) "Query-based learning applied to partially trained multiplayer perceptrons." *IEEE Transactions on Neural Networks.* **2**, 1, 131-136. January, 1991. IEEE Press.

53. Jacobs, R.A., Jordan, M. I., Nowlan, S.J. and Hinton, G.E. (1991). "Adaptive Mixtures of Local Experts." *Neural Computation*, **3**, 79-87.

54. Jacobs, R.A.(1995). "Methods for combining experts' probability assessments." *Neural Computation,* **7**, 867-888.

55. Kashyap, R.L. and Maiyuran, S. (1993) "Robust regression and outlier set estimation using likelihood reasoning." *Technical Report TR-EE 93-8*. School of Electrical Engineering. Purdue University. West Lafayette, Indian 47907-1285.

56. Kindermann, J., Paass, G. and Weber, F. (1995). "Query construction for neural networks using the bootstrap." *Tech. Report*, (GMD) German National Research Centre for Information Technology.

57. Khuri, A.I. and Cornell, J.A. (1987). **Response Surfaces (Designs and Analyses).** Marcel Dekker, Inc., New York.

58. Knight, J.C. and Leveson, N.G.(1986) " An experimental evolution of independence in multiversion programming." *Transactions on Software Engineering*, **SE-12**, 96-109.

59. Krogh, A. and Vedelsby, J. (1995) "Neural Network Ensemble, Cross Validation, and Active Learning." *Advances in Neural Information Processing Systems 7*, MIT press, Cambridge MA.

60. Krogh, A. and Sollich, P. (1996) "Statistical Mechanics of Ensemble Learning." NORDITA preprint 96/24 S (Submitted to Physical Review E).

61. Kurtz, S.A. and Smith, C.H. (1989) "On the role of search for learning." *Proc. COLT'89* (*(Second Annual Workshop on Computational Learning Theory.)*

62. LeBlanc, M. and Tibshirani, R. (1993) "Combining estimates in regression and classification." Paper available from ftp site: ustat.toronto.edu.

63. Levin, E., Tishby, N. and Solla, S.A. (1989). "A statistical approach to learning and generalisation." *Proc. COLT'89 (Second Annual Workshop on Computational Learning Theory.)*

64. Lewis, F., David, D. and William, A. G. (1994). Proc. of ACM SIGIR'94.

65. Lindley, D.V. (1956). "On a measure of the information provided by an experiment." *Ann. Math. Statist.* **27**, 986-1005

66. Littlewood, B. and Miller, D.R. (1989) "Conceptual modelling of coincident failures in multiversion software." *IEEE Transactions on Software Engineering*, **15**, pp.1596-1614.

67. Liu, Yong (1994) Communication broadcast over the *connectionists* electronic mailing list.

68. Luttrell, S.P. (1985). " The use of transformation in the design of data sampling schemes for inverse problems." *Inverse Prob.* **1**, 199-218.

69. Maass, W. (1994). "Neural nets with superlinear vc-dimension." Extended abstract appears in *Proc. of the Int. Conf. on Artificial Neural Nets,* Sorrento (Italy) . To appear in *Neural Computation.*

70. Mackay, D.J.C. (1991). "Bayesian Interpolation." *Neural Comp.* **4**, 415-447.

71. MacKay, D.J.C.(1992) "The evidence framework applied to classification networks." *Neural Computation.* **4**, 720-736. MIT Press.

72. Mackay, D.J.C. (1992b). "A practical Bayesian framework for backprop networks." *Neural Comp.* **4**, 448-472.

73. Mackay, D.J.C. (1992d). "Information-based objective functions for active data selection." *Neural Computation*, **4**(4): 589-603.

74. McCaffrey, D.F. and Gallant, A.R. (1994) "Convergence rates for single hidden layer feedforward networks." *Neural Networks.* **7**, 1, pp.147-158.

75. Meir, R. (1994) " Bias, variance and the combination of estimators; the case of linear least squares." *Preprint (In Neuroprose*), Technion, Haifa, Israel.

76. Myers, R. H. and Khuri, A.I. (1989). " Response Surface Methodology: 1966 - 1988." *Technometrics.* **31**, 2.

77. Muller, Hans-Georg. (1984) "Optimal designs for nonparametric kernel regression." *Statistics and Probability Letter 2.* pp285-290.

78. Nilsson, N.J. (1965) " Learning Machines: Foundations of Trainable Pattern-Classifying Systems." NY: McGraw Hill.

79. Niyogi, P. (1995). "Active learning by sequential optimal recovery." *A.I. Memo No. 1514*, MIT AI Laboratory.

80. Paass, G. (1994). "Assessing and Improving Neural Network Predictions by the Bootstrap Algorithm." German National Research Centre for Computer Science, D-5205 Sankt Augustin, Germany.

81. Paass, G. and Kindermann, J. (1995). "Bayesian query construction for neural network models." *NIP-7 Proceedings.*

82. Parmanto, B., Munro, P. W. and Doyle, H. R. (1996) " Reducing Variance of Committee Prediction with Resampling Techniques." *Connection Science, Vol.8, Nos* 3 & 4, 405-425.

83. Paul, R. (1994)) Communication broadcast over the *Connectionists* Electronic mailing list.

84. Perrone, M. P. and Cooper, L. N. (1993). "When Networks Disagree: Ensemble Methods for Hybrid neural Networks." *Neural Networks for Speech and Image Processing. London*: Chapman and Hall.

85. Platt, J. (1991). "A Resource-Allocating Neural Network for Function Interpolation." Neural Computation. **3**,2, pp213-225.

86. Plutowski, M. and White, H. (1991). " Active selection of training examples for network learning in noiseless environments." *Technical Report CS91-180*, Department of Computer Science and Engineering, University of California, San Diego.

87. Plutowski, M. and White, H. (1993). "Selecting concise training sets from clean data." *IEEE Trans. on Neural Networks*, **4**, 305-318.

88. Plutowski, M. (1994). "Selecting Training Exemplars for Neural Network Learning", PhD dissertation, Department of Computer Science and Engineering, University of California, San Diego, USA.

89. RayChaudhuri, T. and Hamey, L. G. C. (1995). "An Algorithm for Active Data Collection for Learning --- Feasibility Study with Neural Networks." *Macquarie University Technical Report No. 95-173C.*

90. Raviv, Y. and Intrator, T. (1996) "Bootstrapping with Noise; An Effective regularization Technique." *Connection Science, Vol.8, Nos 3 & 4* , 355-372.

91. Riedmiller, M. (1994). "Advanced supervised learning in multi-layer perceptrons - from backpropagation to adaptive learning algorithms." *Int. J. of Computer Standards and Interfaces*, Special Issues on Neural Networks. 5.

92. Rogova, G. (1994) "Combining the results of several neural network classifiers." *Neural Networks*, 7, 777-781.

93. Rumelhart, D.E. and McClelland, J.L. (1986) "On learning the past tense of English verbs." In Rumelhart, D.E., McClelland, J.L. and PDP research Group (Eds), *Parallel*

Distributed Processing: Explorations in the Microstructure of Cognition. Cambridge, MA: MIT Press.

94. Seber, G.A.F and Wild, C.J. (1989). **Nonlinear Regression**. Wiley, New York.

95. Seung, H.S., Opper, M. and Sompolinsky, H. (1992) "Query by committee." In *proceedings of the Fifth Workshop on Computational Learning Theory*, pp.287-294, San Mateo, CA.

96. Sollich, P and Sadd, D. (1994). Learning from queries for maximum information gain in imperfectly learnable problems. *NIPS-6 Proceedings.*

97. Sollich, P.(1994). "Query construction, entropy, and generalisation in neural-network models." *Physical Review E.* Vol.49, No.5. Department of Physics, University of Edinburgh, UK.

98. Stone, M. (1978) "Cross-Validation: A Review." *Mathematishe Operationsforschung Statistishen, Serie Statistics* **9**, 127-139.

99. Sung, K.K and Niyogi, P. (1996). "Active learning for function approximation." *Advances in Neural Information Processing Systems 7*, MIT press, Cambridge MA.

100. Thrun, S.B. and Moller, K. (1993). "Active exploration in Dynamic environments." School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213.

101. Tibshirani, R. (1995). "A comparison of some error estimates for neural network models." *Tech. Report,* Department of Statistics, University of Toronto, October 1995.

102. Tresp,V. and Taniguchi, M. (1995). "Combining estimators using non-constant weighting functions." *NIPS-7 Proceedings.*

103. Tumer, K. and Ghosh, J. (1995) "Theoretical foundations of linear and order statistics combiners for neural pattern classifiers." *Technical Report TR-95-02-98*, The Computer and Vision Research Center, University of Texas, Austin.

104. Tumer, K. and Ghosh, J. (1996). "Error Correlation and Error Reduction in Ensemble Classifiers." *Connection Science, Vol.8, Nos 3 & 4* , 385-404.

105. Valiant, L.G. (1984) "A theory of the learnable." *Comm. of the ACM.* **27**, 11.

106. White, H. (1990). "Learning in Artificial neural networks: A Statistical Perspective." *Neural Computation*, **1**, 4, pp425-464. MIT Press, Cambridge, MA.

107. White, H. (1993) "Estimation, Inference, and Specification Analysis." *Manuscript.*

108.Wolpert, D.H. (1992) "Stacked generalisation." *Neural Networks*, **5**, 241-259.

109.Xu, L., Krzyzak, A. and Suen, C.Y. (1992). "Methods of combining multiple classifiers and their applications to handwriting recognition." *IEEE Transactions on Systems, Man, Cybernetics*, **22**, 418-435.

110.Yamada, K. and Garrison C. (1994) *Working Paper*. Department of Computer Science and Engineering. University of California, San Diego. La Jolla, California. 92093.

111.Yakowitz, S., and Lugosi, E. (1990). " Random search in the presence of noise, with application to machine learning." *Society for Industrial and Applied Mathematics Journal on Scientific and Statistical Computing*. **11**, 4, pp. 702-712.

112.Zhang, B.T. (1994). "Accelerated learning by active example selection." *International Journal of Neural Systems*, **5**(1), 67-75.

113.Zhilkin, P. A. and Somorjai, F. (1996). "Application of Several Methods of Classification Fusion to Magnetic Resonance Spectra." *Connection Science, Vol.8, Nos 3 & 4* , 427-442